nexwave»

Nexwave

J2EE Interview

Q & A

What makes J2EE suitable for distributed multitiered Applications?
The J2EE platform uses a multitiered distributed application model. Application logic is divided into components according to function, and the various application components that make up a J2EE application are installed on different machines depending on the tier in the multitiered J2EE environment to which the application component belongs. The J2EE application parts are:

- Client-tier components run on the client machine.
- Web-tier components run on the J2EE server.
- Business-tier components run on the J2EE server.
- Enterprise information system (EIS)-tier software runs on the EIS server.

What is J2EE?
J2EE is an environment for developing and deploying enterprise applications. The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitiered, web-based applications.

What are the components of J2EE application?
- A J2EE component is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and communicates with other components. The J2EE specification defines the following J2EE components:

1. Application clients and applets are client components.
2. Java Servlet and JavaServer Pages technology components are web components.
3. Enterprise JavaBeans components (enterprise beans) are business components.
4. Resource adapter components provided by EIS and tool vendors.

What do Enterprise JavaBeans components contain?
Enterprise JavaBeans components contains Business code, which is logic that solves or meets the needs of a particular business domain such as banking, retail, or finance, is handled by enterprise beans running in the business tier. All the business code is contained inside an Enterprise Bean which receives data from client programs, processes it (if necessary), and sends it to the enterprise information system tier for storage. An enterprise bean also retrieves data from storage, processes it (if necessary), and sends it back to the client program.

Is J2EE application only a web-based?
No, It depends on type of application that client wants. A J2EE application can be web-based or non-web-based. if an application client executes on the client machine, it is a non-web-based J2EE application. The J2EE application can provide a way for users to handle tasks such as J2EE system or application administration. It typically has a graphical user interface created from Swing or AWT APIs, or a command-line interface. When user request, it can open an HTTP connection to establish communication with a servlet running in the web tier.

Are JavaBeans J2EE components?
No. JavaBeans components are not considered J2EE components by the J2EE specification. They are written to manage the data flow between an application client or applet and components running on the J2EE server or between server components and a database. JavaBeans components written for the J2EE platform have instance variables and get and set methods for accessing the data in the instance variables. JavaBeans components used in this way are typically simple in design and implementation, but should conform to the naming and design conventions outlined in the JavaBeans component architecture.

Is HTML page a web component?
No. Static HTML pages and applets are bundled with web components during application assembly, but are not considered web components by the J2EE specification. Even the server-side utility classes are not considered web components, either.

What can be considered as a web component?

J2EE Web components can be either servlets or JSP pages. Servlets are Java programming language classes that dynamically process requests and construct responses. JSP pages are text-based documents that execute as servlets but allow a more natural approach to creating static content.

What is the container?
Containers are the interface between a component and the low-level platform specific functionality that supports the component. Before a Web, enterprise bean, or application client component can be executed, it must be assembled into a J2EE application and deployed into its container.

What are container services?
A container is a runtime support of a system-level entity. Containers provide components with services such as lifecycle management, security, deployment, and threading.

What is the web container?
Servlet and JSP containers are collectively referred to as Web containers. It manages the execution of JSP page and servlet components for J2EE applications. Web components and their container run on the J2EE server.

What is Enterprise JavaBeans (EJB) container?
It manages the execution of enterprise beans for J2EE applications.
Enterprise beans and their container run on the J2EE server.

What is Applet container?
It manages the execution of applets. Consists of a Web browser and Java Plugin running on the client together.

How do we package J2EE components?
J2EE components are packaged separately and bundled into a J2EE application for deployment. Each component, its related files such as GIF and HTML files or server-side utility classes, and a deployment descriptor are assembled into a module and added to the J2EE application. A J2EE application is composed of one or more enterprise bean,Web, or application client component modules. The final enterprise solution can use one J2EE application or be made up of two or more J2EE applications, depending on design requirements. A J2EE application and each of its modules has its own deployment descriptor. A deployment descriptor is an XML document with an .xml extension that describes a component's deployment settings.

What is a thin client?
A thin client is a lightweight interface to the application that does not have such operations like query databases, execute complex business rules, or connect to legacy applications.

What are types of J2EE clients?
Following are the types of J2EE clients:

- Applets
- Application clients
- Java Web Start-enabled rich clients, powered by Java Web Start technology.
- Wireless clients, based on Mobile Information Device Profile (MIDP) technology.

What is deployment descriptor?
A deployment descriptor is an Extensible Markup Language (XML) text-based file with an .xml extension that describes a component's deployment settings. A J2EE application and each of its modules has its own deployment descriptor. For example, an enterprise bean module deployment descriptor declares transaction attributes and security authorizations for an enterprise bean. Because deployment descriptor information is declarative, it can be changed without modifying the bean source code. At run time, the J2EE server reads the deployment descriptor and acts upon the component accordingly.

What is the EAR file?
An EAR file is a standard JAR file with an .ear extension, named from Enterprise ARchive file. A J2EE application with all of its modules is delivered in EAR file.

What is JTA and JTS?
JTA is the abbreviation for the Java Transaction API. JTS is the abbreviation for the Jave Transaction Service. JTA provides a standard interface and allows you to demarcate transactions in a manner that is independent of the transaction manager implementation. The J2EE SDK implements the transaction manager with JTS. But your code doesn't call the JTS methods directly. Instead, it invokes the JTA methods, which then call the lower-level JTS routines. Therefore, JTA is a high level transaction interface that your application uses to control transaction. and JTS is a low level transaction interface and ejb uses behind the scenes (client code doesn't directly interact with JTS. It is based on object transaction service(OTS) which is part of CORBA.
What is JAXP? - JAXP stands for Java API for XML. XML is a language for representing and describing text-based data which can be read and handled by any program or tool that uses XML APIs. It provides standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and create the appropriate JavaBeans component to perform those operations.
What is J2EE Connector? - The J2EE Connector API is used by J2EE tools vendors and system integrators to create resource adapters that support access to enterprise information systems that can be plugged into any J2EE product. Each type of database or EIS has a different resource adapter. Note: A resource adapter is a software component that allows J2EE application components to access and interact with the underlying resource manager. Because a resource adapter is specific to its resource manager, there is typically a different resource adapter for each type of database or enterprise information system.

What is JAAP?
The Java Authentication and Authorization Service (JAAS) provides a way for a J2EE application to authenticate and authorize a specific user or group of users to run it. It is a standard Pluggable Authentication Module (PAM) framework that extends the Java 2 platform security architecture to support user-based authorization.

What is Java Naming and Directory Service?
The JNDI provides naming and directory functionality. It provides applications with methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes. Using JNDI, a J2EE application can store and retrieve any type of named Java object. Because JNDI is independent of any specific implementations, applications can use JNDI to access multiple naming and directory services, including existing naming and directory services such as LDAP, NDS, DNS, and NIS.

What is Struts?
A Web page development framework. Struts combines Java Servlets, Java Server Pages, custom tags, and message resources into a unified framework. It is a cooperative, synergistic platform, suitable for development teams, independent developers, and everyone between.

How is the MVC design pattern used in Struts framework?
In the MVC design pattern, application flow is mediated by a central Controller. The Controller delegates requests to an appropriate handler. The handlers are tied to a Model, and each handler acts as an adapter between the request and the Model. The Model represents, or encapsulates, an application's business logic or state. Control is usually then forwarded back through the Controller to the appropriate View. The forwarding can be determined by consulting a set of mappings, usually loaded from a database or configuration file. This provides a loose coupling between the View and Model, which can make an application significantly easier to create and maintain. Controller: Servlet controller which supplied by Struts itself; View: what you can see on the screen, a JSP page and presentation components; Model: System state and a business logic JavaBeans.

What is JDBC?
JDBC is a set of Java API for executing SQL statements. This API consists of a set of classes and interfaces to enable programs to write pure Java Database applications.

What are drivers available?
a) JDBC-ODBC Bridge driver
b) Native API Partly-Java driver
c) JDBC-Net Pure Java driver
d) Native-Protocol Pure Java driver

What is the difference between JDBC and ODBC?
a) OBDC is for Microsoft and JDBC is for Java applications.
b) ODBC can't be directly used with Java because it uses a C interface.
c) ODBC makes use of pointers which have been removed totally from Java.
d) ODBC mixes simple and advanced features together and has complex options for simple queries. But JDBC is designed to keep things simple while allowing advanced capabilities when required.
e) ODBC requires manual installation of the ODBC driver manager and driver on all client machines. JDBC drivers are written in Java and JDBC code is automatically installable, secure, and portable on all platforms.
f) JDBC API is a natural Java interface and is built on ODBC. JDBC retains some of the basic features of ODBC.

What are the types of JDBC Driver Models and explain them?
There are two types of JDBC Driver Models and they are:
a) Two tier model and
b) Three tier model
Two tier model: In this model, Java applications interact directly with the database. A JDBC driver is required to communicate with the particular database management system that is being accessed. SQL statements are sent to the database and the results are given to user. This model is referred to as client/server configuration where user is the client and the machine that has the database is called as the server.
Three tier model: A middle tier is introduced in this model. The functions of this model are:
a) Collection of SQL statements from the client and handing it over to the database,
b) Receiving results from database to the client and
c) Maintaining control over accessing and updating of the above.

What are the steps involved for making a connection with a database or how do you connect to a database?
a) Loading the driver : To load the driver, Class. forName() method is used. Class. forName("sun. jdbc. odbc. JdbcOdbcDriver"); When the driver is loaded, it registers itself with the java. sql. DriverManager class as an available database driver.
b) Making a connection with database: To open a connection to a given database, DriverManager. getConnection() method is used. Connection con = DriverManager. getConnection ("jdbc:odbc:somedb", "user", "password");
c) Executing SQL statements : To execute a SQL query, java. sql. statements class is used. createStatement() method of Connection to obtain a new Statement object. Statement stmt = con. createStatement(); A query that returns data can be executed using the executeQuery() method of Statement. This method executes the statement and returns a java. sql. ResultSet that encapsulates the retrieved data: ResultSet rs = stmt. executeQuery("SELECT * FROM some table");
d) Process the results : ResultSet returns one row at a time. Next() method of ResultSet object can be called to move to the next row. The getString() and getObject() methods are used for retrieving column values: while(rs. next()) { String event = rs. getString("event"); Object count = (Integer) rs. getObject("count");

How can you load the drivers?
Loading the driver or drivers you want to use is very simple and involves just one line of code. If, for example, you want to use the JDBC-ODBC Bridge driver, the following code will load it:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Your driver documentation will give you the class name to use. For instance, if the class name is jdbc.DriverXYZ, you would load the driver with the following line of code:

```
Class.forName("jdbc.DriverXYZ");
```

What type of driver did you use in project?
JDBC-ODBC Bridge driver (is a driver that uses native(C language) libraries and makes calls to an existing ODBC driver to access a database engine).

What will Class.forName do while loading drivers?
It is used to create an instance of a driver and register it with the DriverManager. When you have loaded a driver, it is available for making a connection with a DBMS.

How can you make the connection?
To establish a connection you need to have the appropriate driver connect to the DBMS. The following line of code illustrates the general idea:
String url = "jdbc:odbc:Fred";
Connection con = DriverManager.getConnection(url, "Fernanda", "J8?);

What are the types of statements in JDBC?
Statement: to be used createStatement() method for executing single SQL statement
PreparedStatement — To be used preparedStatement() method for executing same SQL statement over and over.
CallableStatement — To be used prepareCall() method for multiple SQL statements over and over.

What is stored procedure?
Stored procedure is a group of SQL statements that forms a logical unit and performs a particular task. Stored Procedures are used to encapsulate a set of operations or queries to execute on database. Stored procedures can be compiled and executed with different parameters and results and may have any combination of input/output parameters.

How to create and call stored procedures?
To create stored procedures: Create procedure procedurename (specify in, out and in out parameters) BEGIN Any multiple SQL statement; END;
To call stored procedures: CallableStatement csmt = con. prepareCall("{call procedure name(?,?)}"); csmt. registerOutParameter(column no. , data type); csmt. setInt(column no. , column name) csmt. execute();

What does setAutoCommit do?
When a connection is created, it is in auto-commit mode. This means that each individual SQL statement is treated as a transaction and will be automatically committed right after it is executed. The way to allow two or more statements to be grouped into a transaction is to disable auto-commit mode:

```
con.setAutoCommit(false);
```

Once auto-commit mode is disabled, no SQL statements will be committed until you call the method commit explicitly.

```
con.setAutoCommit(false);
PreparedStatement updateSales =
        con.prepareStatement( "UPDATE COFFEES SET SALES = ? WHERE COF_NAME LIKE ?");
updateSales.setInt(1, 50); updateSales.setString(2, "Colombian");
updateSales.executeUpdate();
PreparedStatement updateTotal =
        con.prepareStatement("UPDATE COFFEES SET TOTAL = TOTAL + ? WHERE COF_NAME LIKE ?");
updateTotal.setInt(1, 50);
updateTotal.setString(2, "Colombian");
updateTotal.executeUpdate();
con.commit();
con.setAutoCommit(true);
```

How do I retrieve warnings?
SQLWarning objects are a subclass of SQLException that deal with database access warnings. Warnings do not stop the execution of an application, as exceptions do; they simply alert the user that something did not happen as planned. A warning can be reported on a Connection object, a Statement object (including

PreparedStatement and CallableStatement objects), or a ResultSet object. Each of these classes has a getWarnings method, which you must invoke in order to see the first warning reported on the calling object:

```
SQLWarning warning = stmt.getWarnings();
if (warning != null)
{
      System.out.println("n---Warning---n");
      while (warning != null)
      {
            System.out.println("Message: " + warning.getMessage());
            System.out.println("SQLState: " + warning.getSQLState());
            System.out.print("Vendor error code: ");
            System.out.println(warning.getErrorCode());
            System.out.println("");
            warning = warning.getNextWarning();
      }
}
```

How can you move the cursor in scrollable result sets?

One of the new features in the JDBC 2.0 API is the ability to move a result set's cursor backward as well as forward. There are also methods that let you move the cursor to a particular row and check the position of the cursor.

```
Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY);
ResultSet srs = stmt.executeQuery("SELECT COF_NAME, PRICE FROM COFFEES");
```

The first argument is one of three constants added to the ResultSet API to indicate the type of a ResultSet object: TYPE_FORWARD_ONLY, TYPE_SCROLL_INSENSITIVE , and TYPE_SCROLL_SENSITIVE. The second argument is one of two ResultSet constants for specifying whether a result set is read-only or updatable: CONCUR_READ_ONLY and CONCUR_UPDATABLE. The point to remember here is that if you specify a type, you must also specify whether it is read-only or updatable. Also, you must specify the type first, and because both parameters are of type int , the compiler will not complain if you switch the order. Specifying the constant TYPE_FORWARD_ONLY creates a nonscrollable result set, that is, one in which the cursor moves only forward. If you do not specify any constants for the type and updatability of a ResultSet object, you will automatically get one that is TYPE_FORWARD_ONLY and CONCUR_READ_ONLY.

What's the difference between TYPE_SCROLL_INSENSITIVE , and TYPE_SCROLL_SENSITIVE

You will get a scrollable ResultSet object if you specify one of these ResultSet constants.The difference between the two has to do with whether a result set reflects changes that are made to it while it is open and whether certain methods can be called to detect these changes. Generally speaking, a result set that is TYPE_SCROLL_INSENSITIVE does not reflect changes made while it is still open and one that is TYPE_SCROLL_SENSITIVE does. All three types of result sets will make changes visible if they are closed and then reopened:

```
Statement stmt =
      con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
ResultSet srs =
      stmt.executeQuery("SELECT COF_NAME, PRICE FROM COFFEES");
srs.afterLast();
while (srs.previous()){
      String name = srs.getString("COF_NAME");
      float price = srs.getFloat("PRICE");
      System.out.println(name + " " + price);
}
```

How to Make Updates to Updatable Result Sets?

Another new feature in the JDBC 2.0 API is the ability to update rows in a result set using methods in the Java programming language rather than having to send an SQL command. But before you can take advantage of this capability, you need to create a ResultSet object that is updatable. In order to do this, you supply the ResultSet constant CONCUR_UPDATABLE to the createStatement method.

```
Connection con =
      DriverManager.getConnection("jdbc:mySubprotocol:mySubName");
```

```
Statement stmt =
      con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
ResultSet uprs =
      stmt.executeQuery("SELECT COF_NAME, PRICE FROM COFFEES");
```

What is the fastest type of JDBC driver?
Type 4 (JDBC Net pure Java Driver) is the fastest JDBC driver. Type 1 and Type 3 drivers will be slower than Type 2 drivers (the database calls are make at least three translations versus two), and Type 4 drivers are the fastest (only one translation).

Is the JDBC-ODBC Bridge multi-threaded?
No. The JDBC-ODBC Bridge does not support multi threading. The JDBC-ODBC Bridge uses synchronized methods to serialize all of the calls that it makes to ODBC. Multi-threaded Java programs may use the Bridge, but they won't get the advantages of multi-threading.

What is cold backup, hot backup, warm backup recovery?
Cold backup means all these files must be backed up at the same time, before the database is restarted. Hot backup (official name is 'online backup') is a backup taken of each tablespace while the database is running and is being accessed by the users

What is the advantage of denormalization?
Data denormalization is reverse procedure, carried out purely for reasons of improving performance. It maybe efficient for a high-throughput system to replicate data for certain data.

How do you handle your own transaction ?
Connection Object has a method called setAutocommit (boolean flag) . For handling our own transaction we can set the parameter to false and begin your transaction . Finally commit the transaction by calling the commit method.

I have the choice of manipulating database data using a byte[] or a java.sql.Blob. Which has best performance?
java.sql.Blob, since it does not extract any data from the database until you explicitly ask it to. The Java platform 2 type Blob wraps a database locator (which is essentially a pointer to byte). That pointer is a rather large number (between 32 and 256 bits in size) - but the effort to extract it from the database is insignificant next to extracting the full blob content. For insertioninto the database, you should use a byte[] since data has not been uploaded to the database yet. Thus, use the Blob class only for extraction.
Conclusion: use the java.sql.Blob class for extraction whenever you can.

I have the choice of manipulating database data using a String or a java.sql.Clob. Which has best performance?
java.sql.Clob, since it does not extract any data from the database until you explicitly ask it to. The Java platform 2 type Clob wraps a database locator (which is essentially a pointer to char). That pointer is a rather large number (between 32 and 256 bits in size) - but the effort to extract it from the database is insignificant next to extracting the full Clob content. For insertioninto the database, you should use a String since data need not been downloaded from the database. Thus, use the Clob class only for extraction.
Conclusion: Unless you always intend to extract the full textual data stored in the particular table cell, use the java.sql.Clob class for extraction whenever you can.

Do I need to commit after an INSERT call in JDBC or does JDBC do it automatically in the DB?
If your autoCommit flag (managed by the Connection.setAutoCommit method) is false, you are required to call the commit() method - and vice versa.

How can I retrieve only the first n rows, second n rows of a database using a particular WHERE clause ? For example, if a SELECT typically returns a 1000 rows, how do first retrieve the 100 rows, then go back and retrieve the next 100 rows and so on ?

Use the Statement.setFetchSize method to indicate the size of each database fetch. Note that this method is only available in the Java 2 platform. For Jdk 1.1.X and Jdk 1.0.X, no standardized way of setting the fetch size exists. Please consult the Db driver manual.

What does ResultSet actually contain? Is it the actual data of the result or some links to databases? If it is the actual data then why can't we access it after connection is closed?
A ResultSet is an interface. Its implementation depends on the driver and hence ,what it "contains" depends partially on the driver and what the query returns.
For example with the Odbc bridge what the underlying implementation layer contains is an ODBC result set. A Type 4 driver executing a stored procedurethat returns a cursor - on an oracle database it actually returns a cursor in the database. The oracle cursor can however be processed like a ResultSet would be from the client.
Closing a connection closes all interaction with the database and releases any locks that might have been obtained in the process.

What are SQL3 data types?
The next version of the ANSI/ISO SQL standard defines some new datatypes, commonly referred to as the SQL3 types. The primary SQL3 types are:

STRUCT: This is the default mapping for any SQL structured type, and is manifest by the java.sql.Struct type.
REF: Serves as a reference to SQL data within the database. Can be passed as a parameter to a SQL statement. Mapped to the java.sql.Ref type.
BLOB: Holds binary large objects. Mapped to the java.sql.Blob type.
CLOB: Contains character large objects. Mapped to the java.sql.Clob type.
ARRAY: Can store values of a specified type. Mapped to the java.sql.Array type.
You can retrieve, store and update SQL3 types using the corresponding getXXX(), setXXX(), and updateXXX() methods defined in ResultSet interface

How can I manage special characters (for example: " _ ' %) when I execute an INSERT query?
If I don't filter the quoting marks or the apostrophe, for example, the SQL string will cause an error.
The characters "%" and "_" have special meaning in SQL LIKE clauses (to match zero or more characters, or exactly one character, respectively). In order to interpret them literally, they can be preceded with a special escape character in strings, e.g. "\". In order to specify the escape character used to quote these characters, include the following syntax on the end of the query:
{escape 'escape-character'}
For example, the query
SELECT NAME FROM IDENTIFIERS WHERE ID LIKE '_%' {escape '\'}
finds identifier names that begin with an underbar.

What is SQLJ and why would I want to use it instead of JDBC?
SQL/J is a technology, originally developed by Oracle Corporation, that enables you to embed SQL statements in Java. The purpose of the SQLJ API is to simplify the development requirements of the JDBC API while doing the same thing. Some major databases (Oracle, Sybase) support SQLJ, but others do not. Currently, SQLJ has not been accepted as a standard, so if you have to learn one of the two technologies, recommended is JDBC.

How do I insert an image file (or other raw data) into a database?
All raw data types (including binary documents or images) should be read and uploaded to the database as an array of bytes, byte[]. Originating from a binary file,
Read all data from the file using a FileInputStream.
Create a byte array from the read data.
Use method setBytes(int index, byte[] data); of java.sql.PreparedStatement to upload the data.

How can I pool my database connections so I don't have to keep reconnecting to the database?

- you gets a reference to the pool
- you gets a free connection from the pool
- you performs your different tasks
- you frees the connection to the pool

Since your application retrieves a pooled connection, you don't consume your time to connect / disconnect from your data source.

Will a call to PreparedStatement.executeQuery() always close the ResultSet from the previous executeQuery()?
ResultSet is automatically closed by the Statement that generated it when that Statement is closed, re-executed, or is used to retrieve the next result from a sequence of multiple results.

How do I upload SQL3 BLOB & CLOB data to a database?
Although one may simply extract BLOB & CLOB data from the database using the methods of the java.sql.CLOB and java.sql.BLOB, one must upload the data as normal java datatypes. The example below inserts a BLOB in the form of a byte[] and a CLOB in the form of a String into the database

Inserting SQL3 type data [BLOB & CLOB]

```
private void runInsert() {
  try {
      // Log
      this.log("Inserting values ... ");

      // Open a new Statement
      PreparedStatement stmnt = conn.prepareStatement(
        "insert Lobtest (image, name) values (?, ?)");

      // Create a timestamp to measure the insert time
      Date before = new java.util.Date();

      for(int i = 0; i <>
          // Set parameters
          stmnt.setBytes(1, blobData);
          stmnt.setString(2, "i: " + i + ";" + clobData);

          // Perform insert
          int rowsAffected = stmnt.executeUpdate();
      }

      // Get another timestamp to complete the time measurement
      Date after = new java.util.Date();
      this.log(" ... Done!");
      log("Total run time: " + (
        after.getTime() - before.getTime()));

      // Close database resources
      stmnt.close();
  } catch(SQLException ex) {
      this.log("Hmm... " + ex);
  }
}
```

What is the difference between client and server database cursors?
What you see on the client side is the current row of the cursor which called a Result (ODBC) or ResultSet (JDBC). The cursor is a server-side entity only and remains on theserver side.

Are prepared statements faster because they are compiled? if so, where and when are they compiled?
Prepared Statements aren't actually compiled, but they are bound by theJDBC driver. Depending on the driver, Prepared Statements can be a lot faster - if you re-use them. Some drivers bind the columns you request in the SQL statement. When you execute Connection.prepareStatement(), all the columns bindings take place, so the binding overhead does not occur each time you run the Prepared Statement. For additional information on Prepared Statement performance and binding see JDBC Performance Tips on IBM's website.

Is it possible to connect to multiple databases simultaneously? Can one extract/update data from multiple databases with a single statement?

In general, subject, as usual, to the capabilities of the specific driver implementation, one can connect to multiple databases at the same time. At least one driver (and probably others) will also handle commits across multiple connections. Obviously one should check the driver documentation rather than assuming these capabilities.

As to the second part of the question, one needs special middleware to deal with multiple databases in a single statement or to effectively treat them as one database. DRDA (Distributed Relational Database Architecture -- I, at least, make it rhyme with "Gerta") is probably most commonly used to accomplish this.

Oracle has a product called Oracle Transparent Gateway for IBM DRDA and IBM has a product called DataJoiner that make multiple databases appear as one to your application. No doubt there are other products available

Why do I get an UnsupportedOperationException?
JDBC 2.0, introduced with the 1.2 version of Java, added several capabilities to JDBC. Instead of completely invalidating all the older JDBC 1.x drivers, when you try to perform a 2.0 task with a 1.x driver, an UnsupportedOperationException will be thrown. You need to update your driver if you wish to use the new capabilities.

What advantage is there to using prepared statements if I am using connection pooling or closing the connection frequently to avoid resource/connection/cursor limitations?
The ability to choose the 'best' efficiency (or evaluate tradeoffs, if you prefer,) is, at times, the most important piece of a mature developer'sskillset. This is YAA (Yet Another Area,) where that maxim applies. Apparently there is an effort to allow prepared statements to work 'better' with connection pools in JDBC 3.0, but for now, one loses most of the original benefit of prepared statements when the connection is closed. A prepared statement obviously fits best when a statement differing only in variable criteria is executed over and over without closing the statement.

However, depending on the DB engine, the SQL may be cached and reused even for a different prepared statement and most of the work is done by the DB engine rather than the driver. In addition, prepared statements deal withdata conversions that can be error prone in straight ahead, built on the fly SQL; handling quotes and dates in a manner transparent to the developer, for example.

Can I reuse a Statement or must I create a new one for each query?
When using a JDBC compliant driver, you can use the same Statement for any number of queries. However, some older drivers did not always "respect the spec." Also note that a Statement SHOULD automatically close the current ResultSet before executing a new query, so be sure you are done with it before re-querying using the same Statement.

What is a three-tier architecture?
A three-tier architecture is any system which enforces a general separation between the following three parts:

1. Client Tier or user interface
2. Middle Tier or business logic
3. Data Storage Tier

Applied to web applications and distributed programming, the three logical tiers usually correspond to the physical separation between three types of devices or hosts:

What separates one tier from another in the context of n-tiered architecture?
It depends on the application.

In a web application, for example, where tier 1 is a web-server, it may communicate with a tier 2 Application Server using RMI over IIOP, and subsequently tier 2 may communicate with tier 3 (data storage) using JDBC, etc.
Each of these tiers may be on separate physical machines or they may share the same box.
The important thing is the functionality at each tier.
Tier 1 - Presentation - should be concerned mainly with display of user interfaces and/or data to the client browser or client system.

Tier 2 - Application - should be concerned with business logic
Tier 3+ - Storage/Enterprise Systems - should be focused on data persistence and/or communication with other Enterprise Systems.

What areas should I focus on for the best performance in a JDBC application?
These are few points to consider:

- Use a connection pool mechanism whenever possible.
- Use prepared statements. These can be beneficial, for example with DB specific escaping, even when used only once.
- Use stored procedures when they can be created in a standard manner. Do watch out for DB specific SP definitions that can cause migration headaches.
- Even though the jdbc promotes portability, true portability comes from NOT depending on any database specific data types, functions and so on.
- Select only required columns rather than using select * from Tablexyz.
- Always close Statement and ResultSet objects as soon as possible.
- Write modular classes to handle database interaction specifics.
- Work with DatabaseMetaData to get information about database functionality.
- Softcode database specific parameters with, for example, properties files.
- Always catch AND handle database warnings and exceptions. Be sure to check for additional pending exceptions.
- Test your code with debug statements to determine the time it takes to execute your query and so on to help in tuning your code. Also use query plan functionality if available.
- Use proper (and a single standard if possible) formats, especially for dates.
- Use proper data types for specific kind of data. For example, store birthdate as a date type rather than, say, varchar.

How can I insert multiple rows into a database in a single transaction?

```
//turn off the implicit commit

Connection.setAutoCommit(false);
//..your insert/update/delete goes here
Connection.Commit();
a new transaction is implicitly started.
```

How do I convert a java.sql.Timestamp to a java.util.Date?
While Timesteamp extends Date, it stores the fractional part of the time within itself instead of within the Date superclass. If you need the partial seconds, you have to add them back in.
Date date = new Date(ts.getTime() + (ts.getNanos() / 1000000));

What is SQL?
SQL is a standardized language used to create, manipulate, examine, and manage relational databases.

Is Class.forName(Drivername) the only way to load a driver? Can I instantiate the Driver and use the object of the driver?
Yes, you can use the driver directly. Create an instance of the driver and use the connect method from the Driver interface. Note that there may actually be two instances created, due to the expected standard behavior of drivers when the class is loaded.

What's new in JDBC 3.0?
Probably the new features of most interest are:

- Savepoint support
- Reuse of prepared statements by connection pools
- Retrieval of auto-generated keys
- Ability to have multiple open ResultSet objects
- Ability to make internal updates to the data in Blob and Clob objects
- Ability to Update columns containing BLOB, CLOB, ARRAY and REF types

Both java.sql and javax.sql (JDBC 2.0 Optional Package) are expected to be included with J2SE 1.4.

Why do I get the message "No Suitable Driver"?
Often the answer is given that the correct driver is not loaded. This may be the case, but more typically, the JDBC database URL passed is not properly constructed. When a Connection request is issued, the DriverManager asks each loaded driver if it understands the URL sent. If no driver responds that it understands the URL, then the "No Suitable Driver" message is returned.

When I create multiple Statements on my Connection, only the current Statement appears to be executed. What's the problem?
All JDBC objects are required to be threadsafe. Some drivers, unfortunately, implement this requirement by processing Statements serially. This means that additional Statements are not executed until the preceding Statement is completed.

Can a single thread open up mutliple connections simultaneously for the same database and for same table?
The general answer to this is yes. If that were not true, connection pools, for example, would not be possible. As always, however, this is completely dependent on the JDBC driver.

You can find out the theoretical maximum number of active Connections that your driver can obtain via the DatabaseMetaData.getMaxConnections method.

Can I ensure that my app has the latest data?
Typically an application retrieves multiple rows of data, providing a snapshot at an instant of time. Before a particular row is operated upon, the actual data may have been modified by another program. When it is essential that the most recent data is provided, a JDBC 2.0 driver provides the ResultSet.refreshRow method.

What does normalization mean for java.sql.Date and java.sql.Time?
These classes are thin wrappers extending java.util.Date, which has both date and time components. java.sql.Date should carry only date information and a normalized instance has the time information set to zeros. java.sql.Time should carry only time information and a normalized instance has the date set to the Java epoch (January 1, 1970) and the milliseconds portion set to zero.

What's the best way, in terms of performance, to do multiple insert/update statements, a PreparedStatement or Batch Updates?
Because PreparedStatement objects are precompiled, their execution can be faster than that of Statement objects. Consequently, an SQL statement that is executed many times is often created as a PreparedStatement object to increase efficiency.

A CallableStatement object provides a way to call stored procedures in a standard manner for all DBMSes. Their execution can be faster than that of PreparedStatement object.
Batch updates are used when you want to execute multiple statements together. Actually, there is no conflict here. While it depends on the driver/DBMS engine as to whether or not you will get an actual performance benefit from batch updates, Statement, PreparedStatement, and CallableStatement can all execute the addBatch() method.

What is JDO?
JDO provides for the transparent persistence of data in a data store agnostic manner, supporting object, hierarchical, as well as relational stores.

What is the difference between setMaxRows(int) and SetFetchSize(int)? Can either reduce processing time?
setFetchSize(int) defines the number of rows that will be read from thedatabase when the ResultSet needs more rows. The method in the java.sql.Statement interface will set the 'default' value for all the ResultSet derived from that Statement; the method in the java.sql.ResultSet interface will override that value for a specific ResultSet. Since database fetches can be expensive in a networked environment, fetch size has an impact on performance.

setMaxRows(int) sets the limit of the maximum nuber of rows in a ResultSet object. If this limit is exceeded, the excess rows are "silently dropped". That's all the API says, so the setMaxRows method may not help performance at all other than to decrease memory usage. A value of 0 (default) means no limit.

What is DML?
DML is an abbreviation for Data Manipulation Language. This portion of the SQL standard is concerned with manipulating the data in a database as opposed to the structure of a database. The core verbs for DML are SELECT, INSERT, DELETE, UPDATE, COMMIT and ROLLBACK.

What is DDL?
DDL is an abbreviation for Data Definition Language. This portion of the SQL standard is concerned with the creation, deletion and modification of database objects like tables, indexes and views. The core verbs for DDL are CREATE, ALTER and DROP. While most DBMSengines allow DDL to be used dynamically (and available to JDBC), it is often not supported in transactions.

How can I get information about foreign keys used in a table?
DatabaseMetaData.getImportedKeys() returns a ResultSet with data about foreign key columns, tables, sequence and update and delete rules.

How do I disallow NULL values in a table?
Null capability is a column integrity constraint, normally aplied at table creation time. Note that some databases won't allow the constraint to be applied after table creation. Most databases allow a default value for the column as well. The following SQL statement displays the NOT NULL constraint:

```
CREATE TABLE CoffeeTable (
   Type    VARCHAR(25)    NOT NULL,
   Pounds  INTEGER        NOT NULL,
   Price   NUMERIC(5, 2)  NOT NULL
      )
```

How to use Transaction Management in JDBC
Most major RDMBS systems support the concept of transactions. A transaction allows you to group multiple SQL statements together. Using a transaction-aware RDBMS, you can begin a transaction, perform any number of actions, and either commit the results to the database or roll back all of your SQL statements.

A transaction is isolated from the rest of the database until finished. As far as the rest of the database is concerned, everything takes place at once (in other words, transactions are atomic). This means that other users accessing the database will always see a valid view of the data, although not necessarily an up-to-date view. If a user requests a report on widgets sold before your widget sales transaction is completed, the report will not include the most recent sale.

This is done because transactions are linked to connections and, therefore, connections using transactions cannot be shared.

Transaction Control (TCL) statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

COMMIT - save work done
SAVEPOINT - identify a point in a transaction to which you can later roll back
ROLLBACK - restore database to original since the last COMMIT
SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use

Transactions
– By default, after each SQL statement is executed the changes are automatically committed to the database
– Turn auto-commit off to group two or more statements together into a transaction connection.setAutoCommit(false)
– Call commit to permanently record the changes to the database after executing a group of statements
– Call rollback if an error occurs

Useful Connection Methods (for Transactions)

- getAutoCommit/setAutoCommit
  - – By default, a connection is set to auto-commit
  - – Retrieves or sets the auto-commit mode
- commit
  - – Force all changes since the last call to commit to become permanent
  - – Any database locks currently held by this Connection object are released
- rollback
  - – Drops all changes since the previous call to commit
  - – Releases any database locks held by this Connection object

What are ACID Properties

A Transaction is a unit of work performed on the database and treated in a reliable way independent of other transaction. In database transaction processing ACID property refers to the Atomicity, Consistency, Isolation, Durability respectively.

- ✓ **Atomicity**- This property says that all the changes to the data is performed if they are single operation. For example suppose in a bank application if a fund transfer from one account to another account the atomicity property ensures that is a debit is made successfully in one account the corresponding credit would be made in other account.
- ✓ **Consistency**- The consistency property of transaction says that the data remains in the consistence state when the transaction starts and ends. for example suppose in the same bank account, the fund transfer from one account to another account, the consistency property ensures that the total value(sum of both account) value remains the same after the transaction ends.
- ✓ **Isolation**- This property says that, the intermediate state of transaction are hidden/ invisible to another transaction process. Suppose in the the bank application, the isolation property ensures that the fund transfer from one account to another account, the transaction sees the fund transfer in one account or the other account.
- ✓ **Durability**- The Durability says that when the transaction is completed successfully, the changing to the data is persist and is not un-done, even in the event of system failure.

What is Dirty, Non-repeatable and phantom read problems in JDBC?

Dirty Read: Quite often in database processing, we come across the situation wherein one transaction can change a value, and a second transaction can read this value before the original change has been committed or rolled back. This is known as a dirty read scenario because there is always the possibility that the first transaction may rollback the change, resulting in the second transaction having read an invalid value.

While you can easily command a database to disallow dirty reads, this usually degrades the performance of your application due to the increased locking overhead. Disallowing dirty reads also leads to decreased system concurrency.

Non-Repeatable read: One of the ISO-ANSI SQL defined "phenomena" that can occur with concurrent transactions. If one transaction reads a row, then another transaction updates or deletes the row and commits, the first transaction, on re-read, gets modified data or no data. This is an inconsistency problem within a transaction and addressed by isolation levels.

Phantom Read: A "phantom" read occurs when one transaction reads all rows that satisfy a WHERE condition, and a second transaction inserts a row that satisfies that WHERE condition, the first transaction then rereads for the same condition, retrieving the additional "phantom" row in the second read. (from book: JDBC Recipes A Problem-Solution Approach)

What are the JDBC Transaction isolation levels while using Database and explain different types of reads resulting from these isolation levels?

Four types of isolation levels as follows:

1. TRANSACTION_READ_COMMITTED: In this case, all transactions read committed data only, so by using this isolation level, dirty read can be stopped from occuring. As it cannot stop other transactions from changing to adding rows, so non repeatable and phantom reads can occur.
2. TRANSACTION_READ_UNCOMMITTED: As it is not committed, so all three types of reads can occur.
3. TRANSACTION_REPEATABLE_READ: In this isolation level, dirty and non repeatable reads cannot occur, but phantom read can occur.

4. TRANSACTION_SERIALIZABLE: Transaction serializable type of Isoclation Level can stop all three types of reads from occurring, but at the expense of performance, as at a time only on transaction can have access to a table as one point of time. This makes all other transactions to wait, so this is not good performance oriented option.

What isolation level is used by the DBMS when inserting, updating and selecting rows from a database?

The answer depends on both your code and the DBMS. If the program does not explicitly set the isolation level, the DBMS default is used. You can determine the default using DatabaseMetaData.getDefaultTransactionIsolation() and the level for the current Connection with Connection.getTransactionIsolation(). If the default is not appropriate for your transaction, change it with Connection.setTransactionIsolation(int level).

What is servlet?

Servlets are modules that extend request/response-oriented servers, such as java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database.

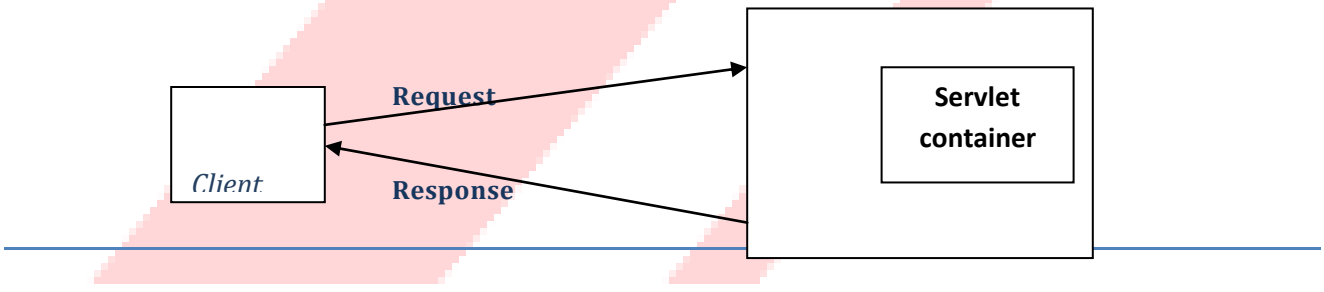
What is Servlet API?

Any server that support the Servlet API (or others like JRun or ServletExec) may run Servlets. The Servlet API is a specification developed by Sun that defines the classes and interfaces used to create and execute servlets. Common examples which can support servlets are FTP, Telnet, mail and news servers.
Any software developed as per the specifications of Servlet API can run the servlets. Examples of software that can run servlets are

1. Servlet Development Kit(SDK) from Sun
2. JRun from Live Software
3. ServletExec from Atlanta Communications and many more. That is SDK, JRun and ServletExec are written as per the specifications of Servlet API.

How Servlets work? OR What is Servlets architecture?

Servlets are framed to implement the request / response paradigm(pattern). For example, when a browser sends a request to the server, the server may forward the request to a servlet (servlet is a software working inside the server itself). At this point, the servlet process the request(through the database access or any other means) and constructs an appropriate response(answer) that is returned(usually in HTML format) to the client. Servlet container (like JSP container for all JSP programs) is the context area in which all servlets of the server works.



What are the classes and interfaces for servlets?

There are two packages in servlets and they are javax. servlet and java.servlet.

What is the basic structure of Servlet ?

The basic structure of a Servlet is nothing but the structure of any Java application. You will notice some new methods, interfaces and exceptions. Your simple Java program becomes a servlet when it extends either GenericServlet or its subclass HTTPServlet(like a Java program becomes applet when extends Applet). By doing so, your Java program (now, it is servlet), attains some default functionality of a servlet. The method that we generally override, is service( ) method. service( ) method (like init( ) of Applet) is called by the

server automatically when a request is made by the client. service( ) method is a general all-purpose method which can be replaced, at the fancy of the developer, by more specific methods like doGet( ) or doPost( ) etc.

What is the Lifecylce of an Servlet ?

Just like an applet got a life cycle, Servlet too got a life cycle. The process of creating, invoking and destroying a Servlet is known as Servlet Lifecycle. These methods that involve life cycle are init( ), service( ) and destroy( ).

- init( ) method is called first time when the servlet is first loaded. It is similar to a class constructor (or init( ) of an Applet)where initialization code is guaranteed to run.
- service( ) method may be overriden by servlets to obtain custom functionality. The method that is automatically called by the server in response to a client request is called service( ) method(like paint( ) method is called when applet is created).
- destroy( ) method is executed when the servlet is unloaded. It is used to free any resources held by the servlet(like destroy( ) method of Applet).

What is the difference between an applet and a servlet?

a) Servlets are to servers what applets are to browsers.
b) Applets must have graphical user interfaces whereas servlets have no graphical user interfaces.

What is the difference between doPost and doGet methods?

a) doGet() method is used to get information, while doPost() method is used for posting information.
b) doGet() requests can't send large amount of information and is limited to 240-255 characters. However, doPost()requests passes all of its data, of unlimited length.
c) A doGet() request is appended to the request URL in a query string and this allows the exchange is visible to the client, whereas a doPost() request passes directly over the socket connection as part of its HTTP request body and the exchange are invisible to the client.

What is the life cycle of a servlet?

Each Servlet has the same life cycle:
a) A server loads and initializes the servlet by init () method.
b) The servlet handles zero or more client's requests through service() method.
c) The server removes the servlet through destroy() method.

Who is loading the init() method of servlet?

Web server

What are the different servers available for developing and deploying Servlets?

a) Java Web Server b) JRun g) Apache Server h) Netscape Information Server i) Web Logic

How many ways can we track client and what are they?

The servlet API provides two ways to track client state and they are:
a) Using Session tracking and
b) Using Cookies.

What is session tracking and how do you track a user session in servlets?

Session tracking is a mechanism that servlets use to maintain state about a series requests from the same user across some period of time. The methods used for session tracking are:
a) User Authentication - occurs when a web server restricts access to some of its resources to only those clients that log in using a recognized username and password.
b) Hidden form fields - fields are added to an HTML form that are not displayed in the client's browser. When the form containing the fields is submitted, the fields are sent back to the server.
c) URL rewriting - every URL that the user clicks on is dynamically modified or rewritten to include extra information. The extra information can be in the form of extra path information, added parameters or some custom, server-specific URL change.
d) Cookies - a bit of information that is sent by a web server to a browser and which can later be read back from that browser.

e) HttpSession- places a limit on the number of sessions that can exist in memory. This limit is set in the session. maxresidents property.

What is HTTP protocol ? Distinguish between get and post methods?
HTML forms creates an interactive environment by providing specific formats like order entry forms. Forms can enclose GUI components like text fields, checkboxes etc.

METHOD attribute specifies the protocol to be used when the client sends the data to the server. There are two choices: GET or POST.

The following is the way to write a method attribute:

```
< FORM  METHOD = "POST"    ACTION = http://localhost:8080/TimeServlet >
< / FORM >
```

Differences between GET and POST methods:
GET method is generally the default method for browsers to submit the data in the form. GET uses HTTP protocol. The GET method is not very secure since the data input appears in the URL. GET method is deprecated. Even though it is deprecated, but still it is the default method.

POST method is more appropriate than GET method. The POST method transmits all form input information immediately after the requested URL. In other words, once the server has received a request from a form using POST, it knows to continue "listening" for the rest of the information. In some sense, this method requires two contacts to make to the Web server. The GET method required only one, because the method comes with the data to use right in the request. The encoding of the form data is handled in the same general way as the GET method by default; spaces become + signs and other characters are encoded in the URL fashion. (? append URL with query string and # to separate field to field

The benefit of using the POST method is that a large amount of data can be submitted this way because the form contents are not in the URL. It is possible to send the contents of files using this method. Encoding POST is same that of GET. It is possible to change the encoding by using the attribute ENCTYPE.

What is Server-Side Includes (SSI)?
Server-Side Includes allows embedding servlets within HTML pages using a special servlet tag. In many servlets that support servlets, a page can be processed by the server to include output from servlets at certain points inside the HTML page. This is accomplished using a special internal SSINCLUDE, which processes the servlet tags. SSINCLUDE servlet will be invoked whenever a file with an. shtml extension is requested. So HTML files that include server-side includes must be stored with an . shtml extension.

What are cookies and how will you use them?
Cookies are a mechanism that a servlet uses to have a client hold a small amount of state-information associated with the user.
a) Create a cookie with the Cookie constructor: public Cookie(String name, String value)
b) A servlet can send a cookie to the client by passing a Cookie object to the addCookie() method of HttpServletResponse: public void HttpServletResponse. addCookie(Cookie cookie)
c) A servlet retrieves cookies by calling the getCookies() method of HttpServletRequest: public Cookie[ ] HttpServletRequest. getCookie().

Is it possible to communicate from an applet to servlet and how many ways and how?
Yes, there are three ways to communicate from an applet to servlet and they are:

a) HTTP Communication(Text-based and object-based)
b) Socket Communication
c) RMI Communication

What is connection pooling?
With servlets, opening a database connection is a major bottleneck because we are creating and tearing down a new connection for every page request and the time taken to create connection will be more. Creating a connection pool is an ideal approach for a complicated servlet. With a connection pool, we can duplicate only

the resources we need to duplicate rather than the entire servlet. A connection pool can also intelligently manage the size of the pool and make sure each connection remains valid. A number of connection pool packages are currently available. Some like DbConnectionBroker are freely available from Java Exchange Works by creating an object that dispenses connections and connection Ids on request. The ConnectionPool class maintains a Hastable, using Connection objects as keys and Boolean values as stored values. The Boolean value indicates whether a connection is in use or not. A program calls getConnection() method of the ConnectionPool for getting Connection object it can use; it calls returnConnection() to give the connection back to the pool.

Why should we go for interservlet communication?
Servlets running together in the same server communicate with each other in several ways. The three major reasons to use interservlet 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)

Is it possible to call servlet with parameters in the URL?
Yes. You can call a servlet with parameters in the syntax as (?Param1 = xxx || m2 = yyy).

What is the difference between forward and sendRedirect?
When you invoke a forward request, the request is sent to another resource on the server, without the client being informed that a different resource is going to process the request. This process occurs completly with in the web container. When a sendRedirtect method is invoked, it causes the web container to return to the browser indicating that a new URL should be requested. Because the browser issues a completly new request any object that are stored as request attributes before the redirect occurs will be lost. This extra round trip a redirect is slower than forward.

What is Servlet chaining?
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 piped to the next servlet's input. This process continues until the last servlet is reached. Its output is then sent back to the client.

What is load-on-startup tag in Servlets ?
The load-on-startup element indicates that this servlet should be loaded (instantiated and have its init() called) on the startup of the web application. The optional contents of these element must be an integer indicating the order in which the servlet should be loaded. If the value is a negative integer, or the element is not present, the container is free to load the servlet whenever it chooses. If the value is a positive 128 integer or 0, the container must load and initialize the servlet as the application is deployed. The container must guarantee that servlets marked with lower integers are loaded before servlets marked with higher integers. The container may choose the order of loading of servlets with the same load-on-start-up value.

How do servlets handle multiple simultaneous requests?
The server has multiple threads that are available to handle requests. When a request comes in, it is assigned to a thread, which calls a service method (for example: doGet(), doPost() and service()) of the servlet. For this reason, a single servlet object can have its service methods called by many threads at once.

What is the difference between TCP/IP and UDP?
TCP/IP is a two-way communication between the client and the server and it is a reliable and there is a confirmation regarding reaching the message to the destination. It is like a phone call. UDP is a one-way communication only between the client and the server and it is not a reliable and there is no confirmation regarding reaching the message to the destination. It is like a postal mail.

What is Inet address?
Every computer connected to a network has an IP address. An IP address is a number that uniquely identifies each computer on the Net. An IP address is a 32-bit number.

What is Domain Naming Service(DNS)?
It is very difficult to remember a set of numbers(IP address) to connect to the Internet. The Domain Naming Service(DNS) is used to overcome this problem. It maps one particular IP address to a string of characters. For example, www. mascom. com implies com is the domain name reserved for US commercial sites, moscom is the name of the company and www is the name of the specific computer, which is mascom's server.

What is URL?
URL stands for Uniform Resource Locator and it points to resource files on the Internet. URL has four components: http://www. address. com:80/index.html, where http - protocol name, address - IP address or host name, 80 - port number and index.html - file path.

What is RMI and steps involved in developing an RMI object?
Remote Method Invocation (RMI) allows java object that executes on one machine and to invoke the method of a Java object to execute on another machine. The steps involved in developing an RMI object are:

a) Define the interfaces
b) Implementing these interfaces
c) Compile the interfaces and their implementations with the java compiler
d) Compile the server implementation with RMI compiler
e) Run the RMI registry
f) Run the application

What is RMI architecture?
RMI architecture consists of four layers and each layer performs specific functions:

a) Application layer - contains the actual object definition.
b) Proxy layer - consists of stub and skeleton.
c) Remote Reference layer - gets the stream of bytes from the transport layer and sends it to the proxy layer.
d) Transportation layer - responsible for handling the actual machine-to-machine communication.

what is UnicastRemoteObject?
All remote objects must extend UnicastRemoteObject, which provides functionality that is needed to make objects available from remote machines.

Explain the methods, rebind() and lookup() in Naming class?
rebind() of the Naming class(found in java. rmi) is used to update the RMI registry on the server machine. Naming. rebind("AddSever", AddServerImpl); lookup() of the Naming class accepts one argument, the rmi URL and returns a reference to an object of type AddServerImpl.

What is a Java Bean?
A Java Bean is a software component that has been designed to be reusable in a variety of different environments.

What is a Jar file?
Jar file allows to efficiently deploying a set of classes and their associated resources. The elements in a jar file are compressed, which makes downloading a Jar file much faster than separately downloading several uncompressed files. The package java. util. zip contains classes that read and write jar files.

What is BDK?
BDK, Bean Development Kit is a tool that enables to create, configure and connect a set of set of Beans and it can be used to test Beans without writing a code.

What is JSP?
JSP is a dynamic scripting capability for web pages that allows Java as well as a few special tags to be embedded into a web file (HTML/XML, etc). The suffix traditionally ends with .jsp to indicate to the web server that the file is a JSP files. JSP is a server side technology - you can't do any client side validation with it. The advantages are:

a) The JSP assists in making the HTML more functional. Servlets on the other hand allow outputting of HTML but it is a tedious process.
b) It is easy to make a change and then let the JSP capability of the web server you are using deal with compiling it into a servlet and running it.

What are JSP scripting elements?
JSP scripting elements lets to insert Java code into the servlet that will be generated from the current JSP page. There are three forms:
a) Expressions of the form <%= expression %> that are evaluated and inserted into the output,
b) Scriptlets of the form<% code %>that are inserted into the servlet's service method, and
c) Declarations of the form <%! Code %>that are inserted into the body of the servlet class, outside of any existing methods.

What are JSP Directives?
A JSP directive affects the overall structure of the servlet class. It usually has the following form:<%@ directive attribute="value" %> However, you can also combine multiple attribute settings for a single directive, as follows:<%@ directive attribute1="value1? attribute 2="value2? . . . attributeN ="valueN" %> There are two main types of directive:
page, which lets to do things like import classes, customize the servlet superclass, and the like; and include, which lets to insert a file into the servlet class at the time the JSP file is translated into a servlet

What are Predefined variables or implicit objects?
To simplify code in JSP expressions and scriptlets, we can use eight automatically defined variables, sometimes called implicit objects. They are request, response, out, session, application, config, pageContext, and page.

Explain the life cycle methods of a JSP?
Pre-translated: Before the JSP file has been translated and compiled into the Servlet.
1. **Translated:** The JSP file has been translated and compiled as a Servlet.
2. **Initialized:** Prior to handling the requests in the service method the container calls the jspInit() to initialize the **Servlet.** Called only once per Servlet instance.
3. **Servicing:** Services the client requests. Container calls this method for each request.
4. **Out of service:** The Servlet instance is out of service. The container calls the jspDestroy() method.

What are JSP ACTIONS?
JSP actions use constructs in XML syntax to control the behavior of the servlet engine. You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin. Available actions include:
1. jsp:include - Include a file at the time the page is requested.
2. jsp:useBean - Find or instantiate a JavaBean.
3. jsp:setProperty - Set the property of a JavaBean.
4. jsp:getProperty - Insert the property of a JavaBean into the output.
5. jsp:forward - Forward the requester to a newpage.
6. Jsp: plugin - Generate browser-specific code that makes an OBJECT or EMBED

How do you pass data (including JavaBeans) to a JSP from a servlet?
(1) Request Lifetime: Using this technique to pass beans, a request dispatcher (using either "include" or forward") can be called. This bean will disappear after processing this request has been completed. Servlet: request. setAttribute("theBean", myBean); RequestDispatcher rd = getServletContext(). getRequestDispatcher("thepage. jsp"); rd. forward(request, response); JSP PAGE:<jsp: useBean id="theBean" scope="request" class=". . . . . " />
(2) Session Lifetime: Using this technique to pass beans that are relevant to a particular session (such as in individual user login) over a number of requests. This bean will disappear when the session is invalidated or it times out, or when you remove it. Servlet: HttpSession session = request. getSession(true); session. putValue("theBean", myBean); /* You can do a request dispatcher here, or just let the bean be visible on the next request */ JSP Page:<jsp:useBean id="theBean" scope="session" class=". . . " />

(3) Application Lifetime: Using this technique to pass beans that are relevant to all servlets and JSP pages in a particular app, for all users. For example, I use this to make a JDBC connection pool object available to the various servlets and JSP pages in my apps. This bean will disappear when the servlet engine is shut down, or when you remove it. Servlet: GetServletContext(). setAttribute("theBean", myBean); JSP PAGE:<jsp:useBean id="theBean" scope="application" class=". . . " />

How can I set a cookie in JSP?
response. setHeader("Set-Cookie", "cookie string"); To give the response-object to a bean, write a method setResponse (HttpServletResponse response) - to the bean, and in jsp-file:<% bean. setResponse (response); %>

How can I delete a cookie with JSP?
Say that I have a cookie called "foo, " that I set a while ago & I want it to go away. I simply: <% Cookie killCookie = new Cookie("foo", null); KillCookie. setPath("/"); killCookie. setMaxAge(0); response. addCookie(killCookie); %>

How are Servlets and JSP Pages related?
JSP pages are focused around HTML (or XML) with Java codes and JSP tags inside them. When a web server that has JSP support is asked for a JSP page, it checks to see if it has already compiled the page into a servlet. Thus, JSP pages become servlets and are transformed into pure Java and then compiled, loaded into the server and executed.

How can I enable session tracking for JSP pages if the browser has disabled cookies?
We know that session tracking uses cookies by default to associate a session identifier with a unique user. If the browser does not support cookies, or if cookies are disabled, you can still enable session tracking using URL rewriting. URL rewriting essentially includes the session ID within the link itself as a name/value pair. However, for this to be effective, you need to append the session ID for each and every link that is part of your servlet response. Adding the session ID to a link is greatly simplified by means of of a couple of methods: response.encodeURL() associates a session ID with a given URL, and if you are using redirection, response.encodeRedirectURL() can be used by giving the redirected URL as input. Both encodeURL() and encodeRedirectedURL() first determine whether cookies are supported by the browser; if so, the input URL is returned unchanged since the session ID will be persisted as a cookie. Consider the following example, in which two JSP files, say hello1.jsp and hello2.jsp, interact with each other. Basically, we create a new session within hello1.jsp and place an object within this session. The user can then traverse to hello2.jsp by clicking on the link present within the page.Within hello2.jsp, we simply extract the object that was earlier placed in the session and display its contents. Notice that we invoke the encodeURL() within hello1.jsp on the link used to invoke hello2.jsp; if cookies are disabled, the session ID is automatically appended to the URL, allowing hello2.jsp to still retrieve the session object.

Try this example first with cookies enabled. Then disable cookie support, restart the brower, and try again. Each time you should see the maintenance of the session across pages. Do note that to get this example to work with cookies disabled at the browser, your JSP engine has to support URL rewriting.

```
hello1.jsp
<%@ page session="true" %>
<%
Integer num = new Integer(100);
session.putValue("num",num);
String url =response.encodeURL("hello2.jsp");
%>
<a href='<%=url%>'>hello2.jsp</a>
hello2.jsp
<%@ page session="true" %>
<%
Integer i= (Integer )session.getValue("num");
out.println("Num value in session is "+i.intValue());
```

How can I declare methods within my JSP page?

You can declare methods for use within your JSP page as declarations. The methods can then be invoked within any other methods you declare, or within JSP scriptlets and expressions. Do note that you do not have direct access to any of the JSP implicit objects like request, response, session and so forth from within JSP methods. However, you should be able to pass any of the implicit JSP variables as parameters to the methods you declare. For example:

```
<%!
public String whereFrom(HttpServletRequest req) {
HttpSession ses = req.getSession();
...
return req.getRemoteHost();
}
%>
<%
out.print("Hi there, I see that you are coming in from ");
%>
<%= whereFrom(request) %>
Another Example
file1.jsp:
<%@page contentType="text/html"%>
<%!
public void test(JspWriter writer) throws IOException{
writer.println("Hello!");
}
%>
file2.jsp
<%@include file="file1.jsp"%>
<html>
<body>
<%test(out);% >
</body>
</html>
```

Is there a way I can set the inactivity lease period on a per-session basis?
Typically, a default inactivity lease period for all sessions is set within your JSP engine admin screen or associated properties file. However, if your JSP engine supports the Servlet 2.1 API, you can manage the inactivity lease period on a per-session basis. This is done by invoking the HttpSession.setMaxInactiveInterval() method, right after the session has been created. For example:

```
<%
session.setMaxInactiveInterval(300);
%>
```

would reset the inactivity period for this session to 5 minutes. The inactivity interval is set in seconds.

How can I set a cookie and delete a cookie from within a JSP page?
A cookie, mycookie, can be deleted using the following scriptlet:

```
<%
//creating a cookie
Cookie mycookie = new Cookie("aName","aValue");
response.addCookie(mycookie);
//delete a cookie
Cookie killMyCookie = new Cookie("mycookie", null);
killMyCookie.setMaxAge(0);
killMyCookie.setPath("/");
response.addCookie(killMyCookie);
%>
```

How does a servlet communicate with a JSP page?
The following code snippet shows how a servlet instantiates a bean and initializes it with FORM data posted by a browser. The bean is then placed into the request, and the call is then forwarded to the JSP page, Bean1.jsp, by means of a request dispatcher for downstream processing.

```
public void doPost (HttpServletRequest request, HttpServletResponse response) {
try {
       govi.FormBean f = new govi.FormBean();
       String id = request.getParameter("id");
       f.setName(request.getParameter("name"));
       f.setAddr(request.getParameter("addr"));
       f.setAge(request.getParameter("age"));
       //use the id to compute
       //additional bean properties like info
       //maybe perform a db query, etc.
       // . . .
       f.setPersonalizationInfo(info);
       request.setAttribute("fBean",f);
       getServletConfig().getServletContext().getRequestDispatcher
                ("/jsp/Bean1.jsp").forward(request, response);
       } catch (Exception ex) {
. . .
   }
}
```

The JSP page Bean1.jsp can then process fBean, after first extracting it from the default request scope via the useBean action.

```
jsp:useBean id="fBean" class="govi.FormBean" scope="request"
/ jsp:getProperty name="fBean" property="name"
/ jsp:getProperty name="fBean" property="addr"
/ jsp:getProperty name="fBean" property="age"
/ jsp:getProperty name="fBean" property="personalizationInfo" /
```

How do I have the JSP-generated servlet subclass my own custom servlet class, instead of the default?

One should be very careful when having JSP pages extend custom servlet classes as opposed to the default one generated by the JSP engine. In doing so, you may lose out on any advanced optimization that may be provided by the JSP engine. In any case, your new superclass has to fulfill the contract with the JSP engine by:

Implementing the HttpJspPage interface, if the protocol used is HTTP, or implementing JspPage otherwise Ensuring that all the methods in the Servlet interface are declared final Additionally, your servlet superclass also needs to do the following:

The service() method has to invoke the _jspService() method

The init() method has to invoke the jspInit() method

The destroy() method has to invoke jspDestroy()

If any of the above conditions are not satisfied, the JSP engine may throw a translation error.

Once the superclass has been developed, you can have your JSP extend it as follows:

```
<%@ page extends="packageName.ServletName" %>
```

How can I prevent the word "null" from appearing in my HTML input text fields when I populate them with a resultset that has null values?

You could make a simple wrapper function, like

```
<%!
String blanknull(String s) {
return (s == null) ? "" : s;
}
%>
then use it inside your JSP form, like
<input type="text" name="shoesize" value="<%=blanknull(shoesize)% >" >
```

How can I get to print the stacktrace for an exception occuring within my JSP page?

By printing out the exception's stack trace, you can usually diagonse a problem better when debugging JSP pages. By looking at a stack trace, a programmer should be able to discern which method threw the exception and which method called that method. However, you cannot print the stacktrace using the JSP out implicit variable, which is of type JspWriter. You will have to use a PrintWriter object instead. The following snippet demonstrates how you can print a stacktrace from within a JSP error page:

```
<%@ page isErrorPage="true" %>
```

```
<%
out.println("     ");
 PrintWriter pw = response.getWriter();
 exception.printStackTrace(pw);
out.println(" ");
%>
```

How do you pass an InitParameter to a JSP?

The JspPage interface defines the jspInit() and jspDestroy() method which the page writer can use in their pages and are invoked in much the same manner as the init() and destory() methods of a servlet. The example page below enumerates through all the parameters and prints them to the console.

```
<%@ page import="java.util.*" %>
<%!
ServletConfig cfg =null;
public void jspInit(){
ServletConfig cfg=getServletConfig();
for (Enumeration e=cfg.getInitParameterNames(); e.hasMoreElements();) {
String name=(String)e.nextElement();
String value = cfg.getInitParameter(name);
System.out.println(name+"="+value);
}
}
%>
```