**J2EE questions and answers**

1. **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.
2. **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.
3. **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:
   * **Application clients** and **applets** are client components.
   * **Java Servlet** and **JavaServer Pages** technology components are web components.
   * **Enterprise JavaBeans components** (enterprise beans) are business components.
   * **Resource adapter components** provided by EIS and tool vendors.
4. **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.
5. **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.
6. **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.
7. **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.
8. **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.
9. **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.
10. **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.
11. **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.
12. **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.
13. **What is Applet container?** - Manages the execution of applets. Consists of a Web browser and Java Plugin running on the client together.
14. **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.
15. **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.
16. **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.
17. **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.
18. **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.
19. **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.
20. **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.
21. **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.
22. **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.
23. **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.
24. **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.
25. **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.
26. **Differentiate between .ear,  .jar and .war files?- Answer:** These files are simply zipped file using java jar tool. These files are created for different purposes. Here is the description of these files:  
    **.jar files:** These files are with the .jar extenstion. The .jar files contains the libraries, resources and accessories files like property files.  
    **.war files:** These files are with the .war extension. The war file contains the web application that can be deployed on the any servlet/jsp container. The .war file contains jsp, html, javascript and other files for necessary for the development of web applications.   
    **.ear files:** The .ear file contains the EJB modules of the application.
27. **What is the difference between Session Bean and Entity Bean?** **Answer:** - **Session Bean**: Session is one of the EJBs and it represents a single client inside the Application Server. Stateless session is easy to develop and its efficient. As compare to entity beans session beans require few server resources. A session bean is similar to an interactive session and is not shared; it can have only one client, in the same way that an interactive session can have only one user. A session bean is not persistent and it is destroyed once the session terminates.     
    **Entity Bean:** An entity bean represents persistent global data from the database. Entity beans data are stored into database.
28. **What are the services provided by a container?** **Answer:-** The services provided by container are as follows:  
    a) Transaction management for the bean  
    b) Security for the bean  
    c) Persistence of the bean  
    d) Remote access to the bean  
    e) Lifecycle management of the bean  
    f) Database-connection pooling  
    g) Instance pooling for the bean
29. **What is difference between Java Bean and Enterprise Java Bean?** **Answer: -** Java Bean as is a plain java class with member variables and getter setter methods. Java Beans are defined under JavaBeans specification as Java-Based software component model which includes the features like introspection, customization,  events,  properties and  persistence.   
    Enterprise JavaBeans or EJBs for short are Java-based software components that comply with Java's  EJB specification. EJBs are delpoyed on the EJB container and executes in the EJB container. EJB is not that simple,  it is used for building distributed applications. Examples of EJB are Session Bean, Entity Bean and Message Driven Bean. EJB is used for server side programming whereas java bean is a client side. Bean is only development but the EJB is developed and then deploy on EJB Container.
30. **What is the difference between JTS and JTA? Answer:** In any J2EE application transaction management is one of the most crucial requirements of the application. Given the complexity of today's business requirements, transaction processing occupies one of the most complex segments of enterprise level distributed applications to build, deploy and maintain.  JTS specifies the implementation of a Java transaction manager. JTS specifies the implementation of a Transaction Manager which supports the Java Transaction API (JTA) 1.0 This transaction manager supports the JTA, using which application servers can be built to support transactional Java applications. Internally the JTS implements the Java mapping of the OMG OTS 1.1 specifications. The Java mapping is specified in two packages: org.omg.CosTransactions and org.omg.CosTSPortability. The JTS thus provides a new architecture for transactional application servers and applications, while complying to the OMG OTS 1.1 interfaces internally. This allows the JTA compliant applications to interoperate with other OTS 1.1 complaint applications through the standard IIOP. Java-based applications and Java-based application servers access transaction management functionality via the JTA interfaces. The JTA interacts with a transaction management implementation via JTS. Similarly, the JTS can access resources via the JTA XA interfaces or can access OTS-enabled non-XA resources. JTS implementations can interoperate via CORBA OTS interfaces.  
    The JTA specifies an architecture for building transactional application servers and defines a set of interfaces for various components of this architecture. The components are: the application, resource managers, and the application server. The JTA specifies standard interfaces for Java-based applications and application servers to interact with transactions, transaction managers, and resource managers JTA transaction management provides a set of interfaces utilized by an application server to manage the beginning and completion of transactions. Transaction synchronization and propagation services are also provided under the domain of transaction management.  
    In the Java transaction model, the Java application components can conduct transactional operations on JTA compliant resources via the JTS. The JTS acts as a layer over the OTS. The applications can therefore initiate global transactions to include other OTS transaction managers, or participate in global transactions initiated by other OTS compliant transaction managers.
31. **Can Entity Beans have no create() methods?** **Answer:** Entity Beans can have no create() methods. Entity Beans have no create() method, when entity bean is not used to store the data in the database. In this case entity bean is used to retrieve the data from database.
32. **What are the call back methods in Session bean? Answer:** - Callback methods are called by the container to notify the important events to the beans in its life cycle.  The callback methods are defined in the javax.ejb.EntityBean interface.The callback methods example are ejbCreate(), ejbPassivate(), and ejbActivate().
33. **What is bean managed transaction? Answer:** In EJB transactions can be maintained by the container or developer can write own code to maintain the transaction. If a developer doesn?t want a Container to manage transactions, developer can write own code to maintain the database transaction.
34. **What are transaction isolation levels in EJB?** **Answer:** Thre are four levels of transaction isolation are:  
    \* Uncommitted Read  
    \* Committed Read  
    \* Repeatable Read  
    \* Serializable  
    The four transaction isolation levels and the corresponding behaviors are described below:

|  |  |  |  |
| --- | --- | --- | --- |
| Isolation Level | Dirty Read | Non-Repeatable Read | Phantom Read |
| Read Uncommitted | Possible | Possible | Possible |
| Read Committed | Not possible | Possible | Possible |
| Repeatable Read | Not possible | Not possible | Possible |
| Serializable | Not possible | Not possible | Not possible |