FRONT-END WEB APPLICATION DEVELOPMENT

ASSIGNMENT -1

SUBMITTED BY:

AYUSH KUMAR JHA

SAP ID - 500086400

Enrollment no - R200220083

B.C.A -I.O.T.

Q1.

(a) Discuss the N-tiered architectural model of JAVA EE with diagram.

Ans. Definition of N-Tier Architecture N-tier architecture is also called multi-tier architecture because the software is engineered to have the processing, data management, and presentation functions physically and logically separated. That means that these different functions are hosted on several machines or clusters, ensuring that services are provided without resources being shared and, as such, these services are delivered at top capacity.

Modern enterprise applications have their responsibilities divided over a number of layers. A common architecture is the 3-layer model consisting of presentation, business, and database layers. The presentation layer is responsible for presenting a user interface and handling interactions with the end user. The business layer is responsible for executing business logic. The database layer is responsible for storage of business data; typically a relational database management system is used for this layer. Layering is used throughout computer science for managing complexity where each layer serves a distinct purpose.

Java Platform Enterprise Edition (Java EE) technology provides services to enterprise applications using a multi-layer architecture. Java EE applications are web-enabled and Java based, which means they may be written once and deployed on any container supporting the Java EE standard. An application server is the environment in which the container resides. However, in practice we don't need to distinguish between an application server and a container, so we will use the terms interchangeably. The Java EE specification is supported by commercial vendors such as Sun, IBM, Oracle, BEA Systems as well as open-source ventures such as JBoss.

Java EE presentation layer technologies include servlets, JSP pages, and JSF components. These are developed for a business application then subsequently deployed and run in a web container. A client would interact with the web container either from a browser or an applet. In either case the http or https internet protocol would be used for communication.

Enterprise JavaBeans version 3 (EJB 3) is the technology Java EE version 5 (Java EE 5) provides for the business layer. In Java EE 5 we subdivide the business layer into one layer which is concerned with business processing and a second layer which deals with persistence. In EJB 3 the business processing artifacts are session and message-driven beans. These are developed for a business application and deployed and run in an EJB container. The persistence layer artifact is an entity; this is persisted to the database layer using a persistence provider or persistence engine. The persistence engine implements another specification, the Java Persistence API (JPA). Both EJB 3 and the JPA are specifications for which a number of organizations provide implementations.

Note that our 3-layer model has become 5-layers. The distinction between client/web and business logic/persistence layers is not always made. Consequently we refer to Java EE architecture simply as n-layer or multi-layer. A Java EE container offers many other services such as web services, the Java Messaging Service (JMS), and resource adapters.

Note from the diagram that we can access an EJB directly from a Java SE application, such as Swing, without going through a web container. The Java application can be stand-alone, or can be run from an Application Client Container (ACC). An ACC enables a client executing in its own Java Virtual Machine (JVM) outside the EJB container to access a limited number of Java EE services.

(b) Explain the various factors that can influence the selection of JAVA EE Application server.
Ans.Java EE

- Java EE facilitates development of large scale applications.
- EE is built upon Java SE. It provides functionalities like web applications, and Servlets.
- Java EE is a structured application with a separate client, business, and Enterprise layers.
- It is mainly used for developing web applications.
- Suitable for experienced Java developers who build enterprise-wide applications.
- It provides user authentication.

Ques 2:

(a) Explain the difference between the archive files used in Java EE.

The biggest difference between JAR, WAR and EAR files is the fact that they are targeted toward different environments. An EAR file requires a fully Java Platform, Enterprise Edition (Java EE)- or Jakarta Enterprise Edition (EE)-compliant application server, such as WebSphere or JBoss, to run. A WAR file only requires a Java EE Web Profile-compliant application server to run, and a JAR file only requires a Java installation.
There are also internal restrictions and requirements that apply to EAR, WAR and JAR files. EAR files themselves must have an application.xml file contained within a folder named META-INF. A WAR file requires a web.xml file contained within a WEB-INF folder. Java files have neither of these requirements.
JAR vs. EJB-JAR
The Java EE specification also defines a special type of JAR file that contains only Enterprise JavaBeans (EJB). This file has a .jar extension but contains a special deployment descriptor and is intended to isolate EJB components from other parts of the enterprise application. The Java EE spec also defines a resource adapter archive,

which contains code that bridges an enterprise application to external services, like message queues and databases. These files have a .rar extension.
Microservices and JAR files
The current trend in the software development industry is toward microservices development and away from monolithic applications. As such, there has been a move away from the development and deployment of enterprise applications deployed as EAR files and a move toward the creation of smaller components that are deployed as JAR files.
Modern microservices frameworks, such as Spring Boot and Eclipse MicroProfile, deploy applications as runnable JAR files that can be deployed directly to a software container, such as Docker, and to a container orchestration tool, such as RedHat OpenShift.

(b) Discuss the 5 main categories of logging methods.
Ans.

1. Create a logger

The Logger you create is actually a hierarchy of Loggers, and a . (dot) in the hierarchy indicates a level in the hierarchy. So if you get a Logger for the com.example key, this Logger is a child of the com Logger and the com Logger is child of the Logger for the empty String. You can configure the main logger and this affects all its children.

2. Level

The log levels define the severity of a message. The Level class is used to define which messages should be written to the log.
The following lists the Log Levels in descending order:
- SEVERE (highest)
- WARNING
- INFO
- CONFIG
- FINE
- FINER
- FINEST

In addition to that you also have the levels OFF and ALL to turn the logging off or to log everything.
For example, the following code sets the logger to the info level, which means all messages with severe, warning and info will be logged.

3. Handler

Each logger can have access to several handlers.
The handler receives the log message from the logger and exports it to a certain target.
A handler can be turned off with the setLevel(Level.OFF) method and turned on with setLevel() method.
You have several standard handlers. The following list gives some examples.
- ConsoleHandler: Write the log message to console
- FileHandler: Writes the log message to file

Log levels INFO and higher will be automatically written to the console.

4. Formatter

Each handler’s output can be configured with a formatter
Available formatter

- SimpleFormatter: Generate all messages as text
- XMLFormatter: Generates XML output for the log messages

5. Log Manager

The log manager is responsible for creating and managing the logger and the maintenance of the configuration.
We could set the logging level for a package, or even a set of packages, by calling the LogManager.setLevel(String name, Level level) method. So, for example, we could set the logging level of all loggers to Level.FINE by making this call:
LogManager.getLogManager().getLogger(Logger.GLOBAL_LOGGER_NAME).setLevel(Level.FINE);

Ques 3:

(a) Differentiate between JSPs and Servlets.

Ans.

Servlet

Servlets are faster as compared to JSP, as they have a short response time.

Servlets are Java-based codes.

Servlets are harder to code, as here, the HTML codes are written in Java.

In an MVC architecture, Servlets act as the controllers.

The service() function can be overridden in Servlets.

The Servlets are capable of accepting all types of protocol requests.

Modification in Servlets is a time-consuming and challenging task, as here, one will have to reload, recompile, and then restart the servers.

Servlets require the users to enable the default sessions management explicitly, as Servlets do not provide default session management.

Servlets are hosted and executed on Web Servers.

We need to import all the packages at the top of the Servlets.

JSP

JSP is slower than Servlets, as the first step in the JSP lifecycle is the conversion of JSP to Java code and then the compilation of the code.

JSP are HTML-based codes.

JSPs are easier to code, as here Java is coded in HTML.

In MVC architectures, the JSPs act as a view to present the output to the users.

The service() function cannot be overridden in JSPs.

The JSPs are confined to accept only the HTTP requests.

Modification is easy and faster in JSPs as we just need to refresh the pages.

JSPs provide session management by default.

JSP is compiled in Java Servlets before their execution. After that, it has a similar lifecycle as Servlets.

In JSPs, we can import packages anywhere in the file.

(b) Differentiate between forward and include methods of RequestDispatcher.

Ans. The difference between the two methods is that the forward method will close the output stream after it has been invoked, whereas the include method leaves the output stream open. The include method takes the content from another resource and includes it in the servlet.
1) Both include() and forward() methods are part of RequestDispatcher interface of Servlet API

2) Both methods accept objects of ServletRequest and ServletResponse interface.

3) Both include() and forward() can interact with static and dynamic resource e.g. another Servlet, JSP or HTML files.

Ques 4:

(a) Discuss the JSP page translation procedure.

Each JSP page is internally translated into an equivalent Servlet. Every JSP tag written in the page will be internally converted into an equivalent java code by the containers this process is called translation.

There are two phases are occurs in JSP;

Translation phase
Request processing phase
Translation phase: It is process of converting jsp into an equivalent Servlet and then generating class file of the Servlet.

Request processing phase: It is process of executing service() of jsp and generating response data on to the browser.

When first time a request to a jsp then translation phase occurs first and then after service phase will be executed. From next request to the jsp, only request processing phase is got executed but translation phase is not because jsp is already translated.

In following two cases only translation occurs;

When first request is a given to the jsp.
When a jsp is modified.
If a request is given to the jsp after it has modified then again translation phase is executed first and after that request processing phase will be executed.

(b) Discuss the different HTTP method annotations used for resource methods.