ISO 9001:2008 Certified Institute

JAVA INSTITUTE FOR ADVANCED TECHNOLOGY

Department of Examinations

Course – (Leading To)	Birmingham City BSc (Hons) Se - Top Up
Unit Name	Business Component Development I
Unit Id	JIAT/BCD I
Assignment Id	JIAT/BCD I/EX/01
Assignment Summary	This assignment is based on the technical report and research. The technical report should: 1. Explain the solution you have implemented using appropriate EJB components to solve the given problem. 2. Analyse the efficiency and correctness of your solution.
Duration	1 Week
Submission Via	Online (Student Portal)
Document Format	Document Format (Pdf)

Mathes Nambuhewage Jeral Sandeeptha
200015003010

Phoenix Batch

Colombo Branch

Java Institute for Advanced Technology

Sri Lanka

Task 2

Usage of Enterprise Application Model

The Enterprise Application Model (EAM) is a programming configuration that is utilized to construct complex enterprise applications that can adjust to the necessities of big organizations. The EAM offers a structure for creating applications that can be sent on various servers, disseminated across various areas, and can deal with high volumes of exchanges.

In the recommended answer for using suitable EJB segments for the Smart Energy Management System, the EAM can be utilized to give the accompanying advantages,

First one is scalability. The EAM gives a flexible engineering that can deal with high volumes of data and exchanges. The application can be sent on various servers, which permits it to scale level to meet the necessities of the association.

Second one is availability. The EAM gives a profoundly accessible engineering that guarantees that the application is constantly up and running. The application can be sent across numerous areas, which gives duplication and guarantees that the application is always accessible.

Usage of Containers and Connectors

The proposed solution for the Smart Energy Management System calls for the use of EJB components, Containers, and Connectors, which can bring the following advantages:

Containers:

Containers give an operational atmosphere for EJB components to run. In this answer, the EJB container can be utilized to offer services like transaction control, protection, and life cycle administration to the EJB components.

For instance, the EJB container can handle transactions for the EJB components that grab and transmit the sensor data. The container can guarantee that the transactions are fundamental and that the data is consistently put away in the database.

Connectors:

Connectors provide a standardized approach to link EJB components to outer frameworks, for example, databases and messaging servers. In this solution, the Connector can be utilized to associate the EJB components that catch and transmit the sensor data to the messaging server.

For example, the Connector can be utilized to interface the EJB part that captures the sensor data to the messaging server.

Advantages of Enterprise JavaBeans Component Model

The EJB component model has a number of benefits. These benefits apply to the suggested solution for the Smart Energy Management System that uses the proper EJB components. These are a few of the benefits.

Scalability comes first. Scalable architecture is provided via the EJB component concept. High data and transaction volumes can be handled by this architecture. EJB parts can be installed on numerous servers. This enables them to scale horizontally to meet organizational demands.

Security is the second. The secure architecture offered by the EJB component model guards the application against unauthorized access. To safeguard data, EJB components can be configured to use a variety of security techniques and protocols.

The third is reusable. Developers can reuse components in other applications because to the EJB component model's modular nature. As a result, creating, testing, and deploying the application are made simpler.

Transaction Management ranks fourth. The transactional architecture is provided by the EJB component model. This design guarantees atomic transactions and consistent data. This is therefore quite helpful.

The application's dependability is increased as a result. Additionally, it guarantees that the database is constantly stored with data.

Separation of Concerns is the fifth. The EJB component paradigm enables a division of duties between the infrastructure services, such as transaction management and security, and the application logic. This enhances the application's maintainability and makes it simpler to change and upgrade the application over time.

Overall, the suggested solution's usage of the EJB component model and the proper EJB components for the Smart Energy Management System results in an architecture that is scalable, safe, reusable, and maintainable and can satisfy the needs of large businesses.

Task 3

The difference between dependency injection and initial context lookup.

Both Dependency Injection (DI) and Initial Context Lookup (ICL) can be utilized to gain references to EJB components in the suggested solution for the Smart Energy Management System using appropriate EJB components.

Initial Context Lookup and Dependency Injection very primarily in that they,

Inversion of Control. DI is a type of inversion of control in which the container or framework gains control over the application's creation and management of objects. The creation and injection of an object's dependencies are the responsibility of the container. ICL, in contrast, uses a more conventional methodology in which the program specifically searches for objects in a naming service.

Configuration follows. Configuration is necessary for DI in order to specify an object's dependencies. Annotations or XML files can both be used for this setup. Although the application directly looks up items by name, ICL, in contrast, does not need any setting.

Testing comes next. Due to the ease with which dependencies can be mocked or stubbed, DI makes it simpler to test objects in isolation. ICL, on the other hand, can make testing more challenging due to the application's close relationship with the naming service.

Coupling is the following. Due to dependencies being injected at runtime, DI lessens entanglement between objects. ICL, on the other hand, has the potential to increase coupling between objects because the application makes explicit use of object naming.

The EJB components can be injected into the application logic in the suggested manner, which eliminates coupling and facilitates testing. ICL may be used to look up resources such as database connections, the JNDI naming service, and the EJB container.

Generally, the choice between Initial Context Lookup and Dependency Injection depends on the particular requirements of the application. Both strategies have benefits and drawbacks, and the best strategy should be chosen depending on elements like complexity, maintainability, and testability.

The importance of JMS API for the enterprise-level application

For enterprise-level applications, the Java Messaging Service (JMS) API is crucial, particularly for the suggested approach using the right EJB components for the Smart Energy Management System.

JMS offers a dependable and asynchronous means for messages to be sent and received between various application components. JMS can be used in the Smart Energy Management System to communicate between IoT devices, a messaging server, and an analytical server. The following advantages are offered by the JMS API.

Asynchronous communication comes first.

JMS enables components to interact asynchronously, so the sender need not wait for the receiver to respond. Asynchronous communication enhances system performance by enabling autonomous operation of components.

Dependable message delivery is the second.

JMS offers dependable message delivery, which ensures that messages will reach their intended recipients despite network problems or other disruptions. Data loss can have severe effects in vital applications. It may be like the Smart Energy Management System, so this is crucial.

JMS offers message queuing, which allows for the storage of messages in a queue until they are ready to be processed. By enabling messages to be handled in the order that they are received, this increases the system's scalability.

In the publish/subscribe model, JMS offers a Publish/Subscribe paradigm, enabling the sending of messages to numerous recipients. This is helpful in the Smart Energy Management System because it may be necessary for several devices to receive the same message there.

In general, JMS is a crucial part of enterprise-level programs like the Smart Energy Management System. It offers a dependable and asynchronous method for messages to be sent and received between various application components, which enhances performance, scalability, and dependability.

Usage of the message-driven beans (MDB).

The proposed solution for the Smart Energy Management System incorporates message-driven beans (MDB), which are essential, by leveraging the appropriate EJB components. MDBs are used to consume asynchronous messages from a JMS queue or topic. In this situation, MDBs could be used to achieve the following objectives:

MDBs allow for the consumption of messages from IoT devices. IoT devices send these messages to the messaging server. Information from the devices, like as voltage, frequency, and current data, may be included in the messages.

The communications can be processed by the MDBs and either forwarded to the analytical server or stored in a database.

Messages from the analytical server that provide data such as the total amount of power utilized, the average current, the average voltage, and the average frequency can be consumed by MDBs. The other system components, such as the dashboard, can then receive this information from the MDBs and show it.

Next one is application scaling. By deploying many instances of MDBs, the application can be scaled horizontally. The application can process messages in parallel and increase throughput because each MDB instance can consume messages from the same queue or topic.

The next is that dependable message consumption may be ensured by using MDBs. A message can be redelivered to another instance of the MDB until it is properly processed if an MDB instance is unable to handle it due to a system error or exception. No messages are lost in the system as a result of this.

Message-driven beans offer a dependable and scalable means to ingest messages from the messaging server and analytical server, making them a crucial part of the suggested solution for the Smart Energy Management System.

Task 4

Suitable Session bean type for the implementation of a web application representing the number of times a web page has been accessed. Compare and contrast your implementation among other bean types.

The Stateful Session Bean is an appropriate Session bean type for the construction of a web application that represents the number of times a web page has been accessed.

A Stateful Session Bean can keep track of the number of times a certain client has accessed a web page since it keeps data as instance variables and maintains conversational state with the client. Every time a client accesses a web page, the Stateful Session Bean can update the counter value and return the new value to the client. The counter value is preserved during the communication with the client and is deleted at the conclusion of the conversation thanks to the stateful nature of the bean.

Stateless and Singleton session beans, as well as other types of session beans, are inappropriate for this solution since they do not preserve conversational state with the client. Singleton Session Beans are built for application-wide state that can be accessed by different clients, whereas Stateless Session Beans are designed for processing brief requests that do not require conversational state.

The Stateful Session Bean is more appropriate for representing stateful data in memory than the Entity Bean, which is used for data persistence and represents data stored in a database. Although the Entity Bean can keep track of how many times a web page has been accessed, doing so involves extra overhead when reading and writing data from a database, which may not be necessary for a straightforward counter.

In conclusion, the Stateful Session Bean is the most appropriate type of Session Bean for the implementation of a web application expressing the frequency of access to a web page. It is the best option for keeping and updating the counter value for each client due to its stateful nature and capacity to maintain conversational state with the client.

References

In-text:

(Java 2 Platform, Enterprise Edition (J2EE) Overview, 2023)

Your bibliography:

Anon., 2023. *Java 2 Platform, Enterprise Edition (J2EE) Overview*. [online] Available at: <https://www.oracle.com/java/technologies/appmodel.html> [Accessed 9 March 2023].

In-text:

(Session 11 - The Enterprise JavaBeans (EJB) Server Component .., 2023)

Your bibliography:

Anon., 2023. *Session 11 - The Enterprise JavaBeans (EJB) Server Component .* [online] Available at: <https://www.nyu.edu/classes/jcf/g22.3033-006_sp03/handouts/g22_3033_007_h111.htm> [Accessed 9 March 2023].

In-text:

(Overview of the JMS API - The Java EE 6 Tutorial, 2023)

Your bibliography:

Anon., 2023. *Overview of the JMS API - The Java EE 6 Tutorial*. [online] Available at: <https://docs.oracle.com/javaee/6/tutorial/doc/bncdr.html> [Accessed 9 March 2023].

In-text:

(A Guide to Message Driven Beans in EJB | Baeldung, 2023)

Your bibliography:

Anon., 2023. *A Guide to Message Driven Beans in EJB | Baeldung*. [online] Available at: <https://www.baeldung.com/ejb-message-driven-beans> [Accessed 9 March 2023].