ENTERPRISE JAVA BEANS

* Enterprise Beans are Java EE components that implement Enterprise JavaBeans(EJB) technology.
* Enterprise Java Beans run in the EJB container.
* Although transparent to the developer, the EJB container provides system-level services, such as transactions and security, to its enterprise beans.
* These services enable you to quickly build and deploy enterprise beans, which form the core of transactional Java EE applications.

**When to use EJBs**

* The application must be scalable
* Transactions must ensure data integrity. Enterprise beans support transactions, the mechanisms that manage the concurrent access of shared objects.
* The application will have a variety of clients. With only a few lines of code, remote clients can easily locate enterprise beans. These clients can be thin, various, and numerous.

**Types of Enterprise Beans**

* Session – Performs a task for client; optionally, may implement a web service
* Message-driven – Acts as a listener for a particular messaging type, such as the Java Message Service API

**Message-Driven Bean**

**A** message-driven bean is an enterprise bean that allows a Java EE applications to process messages asynchronously. This type of bean normally acts as a JMS message listener, which is similar to an event listener but receives JMS messages instead of events. Message-driven beans can process JMS messages or other kinds of messages.

**Characteristics:**

Client components do not locate message-driven beans and invoke methods directly on them. Instead, a client accesses a message-driven bean through, for example, JMS by sending messages to the message destination for which the message-driven bean class is the MessageListener. You assign a message-driven bean’s destination during deployment by *using GlassFish Server resources*.

* The execute upon receipt of a single client message
* They are invoked asynchronously
* They are relatively short-lived
* They do not represent directly shared data in the database, by they can access and update the data.
* They can be transaction-aware
* They are stateless.

When a message arrives, the container calls the message-driven bean’s onMessage method to process to the message. The onMessage method normally casts the message to one of the five JMS message types and handles it in accordance with the application’s business logic. The onMessage method can call helper methods or can invoke a session bean to process the information in the message or to store it in a database.

A message can be delivered to a message-driven bean within a transaction context, so all operations within the onMessage method are part of a single transaction. If message processing is rolled back, the message will be redelivered. For more infoation, see Chapter25, A Message-Driven Bean Example and Chapter 44, Transactions.