**EJB-Bean**

We have Stateless, Statefull and Singlton

**Statefull**

Maintains conversational state between client and server.   
Maintains for duration of conversation between client and server  
bean remains available as long as client has not removed it

We use when we need to hold an information about a client

**Stateless**

Maintains the state for duration of method execution.   
The instance of the bean will removed or gone back to pool   
EJB container handling the create and remove method

Benefit of Stateless : Pooling, Scalability and performance

Stateless bean can implement a web Service

We use when a generic can perform the same task for all clients and we do not need to save an state of an object

**Singlton**

Instantiated once per container and exists for life cycle if application   
Similar functionality as Stateless bean except the fact that there is only one instance of it   
Singlton is actually Stateless bean with pool size of 1

Since there is only one instance of this bean so then we can have

* @**PostConstruct** method called with the application starts up and its
* @**PreDestroy** method called when the application shuts down.

It will gives life cycle listener

We use it:

when we need to instantiate something at startup

when we need to share one bean accessed by multiple threads concurrently

**Stateless session façade**

Higher level of façade hierarchy  
Introduce the layer between client and server  
Increase flexibility and maintainability   
Reduce fine-grained methods   
Expose fewer remote interfaces to clients

**JMS**

An Interface implemented by JEE container to provide point-2-point or publish/subscribe behavior

@Resource(name="jms/XATbiCF", type=javax.jms.ConnectionFactory.class)  
@MessageDriven(mappedName="jms/PITCreateCaseRequestQueue")

**MDB**

MDB is a class that implements Message Listener

public class PitCreateCaseMDB implements MessageListener {

void onMessage() {

}

}

**JMS model**

Provides 2 model of messaging :

1. Point-2-point 🡪 Queue messaging so many can send a message to queue but only 1 listener
2. Pulish/Subscribe 🡪 topic

**JMS type**

Message , textMessage, ByeMessages, StreamMessages Object messages,MapMessags

Advatage of JMS

* **Redundancy**. if the transaction fails it can be reprocessed. The messages can also be stored in a database allowing them to continue later on even if the server stops.
* **Asynchronous messaging**.
* **Loose coupling**. The services do not interact directly and only know where the message queue is

What is ActiveMQ ?

It is Apache broker that also that is well integrated with Spring boot.

@Component

public class MessageReceiver {

    static final Logger LOG = LoggerFactory.getLogger(MessageReceiver.class);

    private static final String ORDER\_RESPONSE\_QUEUE = "order-response-queue";

    @Autowired

    OrderService orderService;

    @JmsListener(destination = ORDER\_RESPONSE\_QUEUE)

    public void receiveMessage(final Message<InventoryResponse> message) throws JMSException {

        LOG.info("+++++++++++++++++++++++++++++++++++++++++++++++++++++");

        MessageHeaders headers =  message.getHeaders();

        LOG.info("Application : headers received : {}", headers);

        InventoryResponse response = message.getPayload();

        LOG.info("Application : response received : {}",response);

        orderService.updateOrder(response);

        LOG.info("+++++++++++++++++++++++++++++++++++++++++++++++++++++");

    }

}