ISSN: 2321-7782 (Online)
Volume 2, Issue 8, August 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Enterprise Java Beans (EJB) Types

Abhilasha Raval
M.C.A. Department
IGNOU
Gujarat – India

Abstract: Enterprise JavaBeans facilitates the development of distributed Java applications, providing an object-oriented transactional environment for building distributed multi-tier enterprise components. An EJB is a remote object, which needs the Services of an EJB container in order to execute.

Keywords: EJB, beans, Session Bean, Entity Beans, Message-driven Beans, BMP, CMP.

I. Introduction

EJB is **WORA** (Write Once Run Anywhere). Enterprise JavaBeans takes a high-level approach to building distributed systems. It frees the application developer and enables him/her to concentrate on programming only the business logic while removing the need to write all the “plumbing” code that's required in any enterprise application. For example, the enterprise developer no longer needs to write code that handles transactional behavior, security, connection pooling, networking or threading. The architecture delegates this task to the server vendor. EJBs are distinguished along three main functional roles. Within each primary role, the EJBs are further distinguished according to sub roles.

II. Types of Beans

There are three main types of beans:

- Session Bean
- Entity Beans
- Message-driven Beans

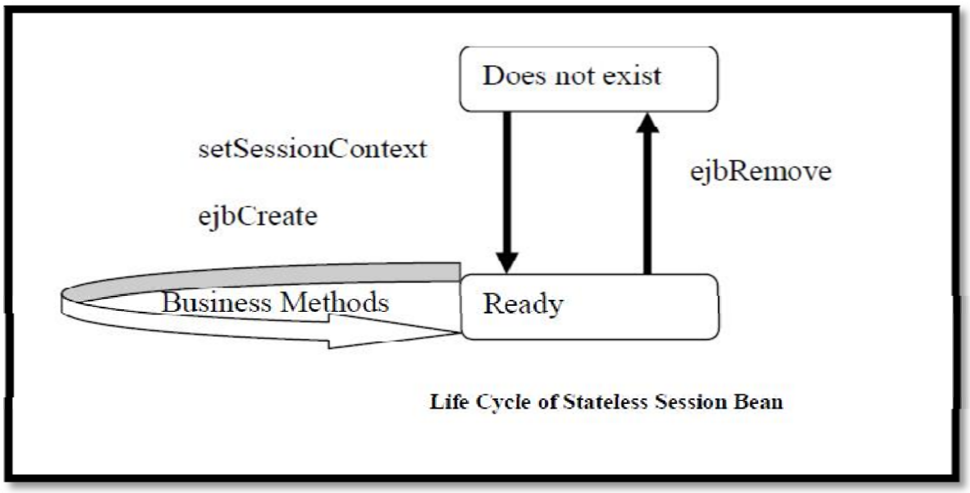
1) Session Bean:

A session EJB is a non persistent object. Its lifetime is the duration of a particular Interaction between the client and the EJB. The client normally creates an EJB, calls methods on it, and then removes it. If, the client fails to remove it, the EJB container will remove it after a certain period of inactivity. There are two types of session beans:

a) **Stateless Session Beans:** A stateless session EJB is shared between a numbers of clients. It does not maintain conversational state. After each method call, the container may choose to destroy a stateless session bean, or recreate it, clearing itself out, of all the information pertaining the invocation of the last method. The algorithm for creating new instance or instance reuse is container specific.

b) **Stateful Session Beans:** A stateful session bean is a bean that is designed to service Business processes that span multiple method requests or transaction. To do this, the stateful bean retains the state for an individual client. If, the stateful bean’s state is changed during method invocation, then, that same state will be available to the same client upon invocation.

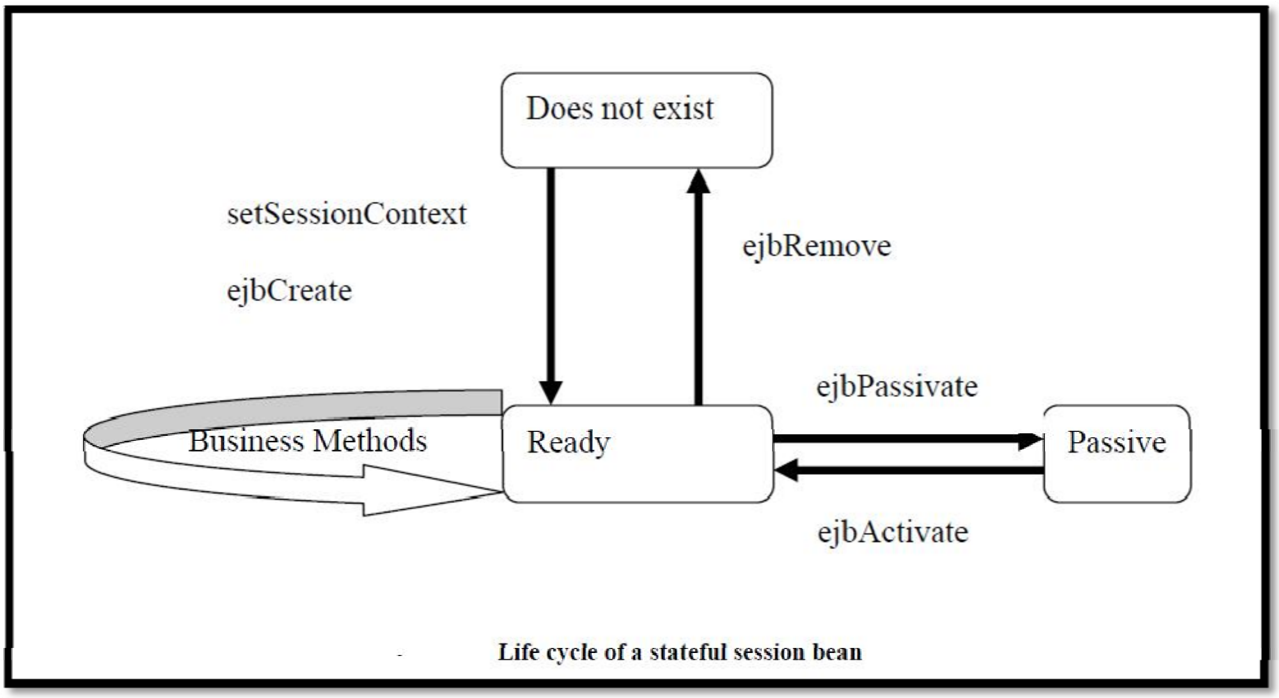
- **Life Cycle of a Stateless Session Bean**



Life Cycle of Stateless Session Bean

- ✓ **Does not exist:** In this state, the bean instance simply does not exist.
- ✓ **Ready state:** When EJB Server is first started, several bean instances are created and placed in the Ready pool. More instances might be created by the container as and when needed by the EJB container.

- **Life Cycle of a Stateful Session Bean**



Life cycle of a stateful session bean

- ✓ **Does not exist:** In this state, the bean instance simply does not exist.
- ✓ **Ready state:** A bean instance in the ready state is tied to a particular client and engaged in a conversation.
- ✓ **Passive state:** A bean instance in the passive state is passivated to conserve resources.

❖ **The use of a Session Bean:**

In general, one should use a session bean if the following circumstances hold:

- ✓ At any given time, only one client has access to the bean instance.
- ✓ The state of the bean is not persistent, existing only for a short period and therefore.
- ✓ The bean implements a web service.

- Stateful session beans are appropriate if, any of the following conditions are true:
  - ✓ The bean's state represents the interaction between the bean and a specific client.
  - ✓ The bean needs to hold information about the client across method invocations.
  - ✓ The bean mediates between the client and the other components of the application,
  - ✓ Presenting a simplified view to the client.
  - ✓ Behind the scenes, the bean manages the work flow of several enterprise beans.

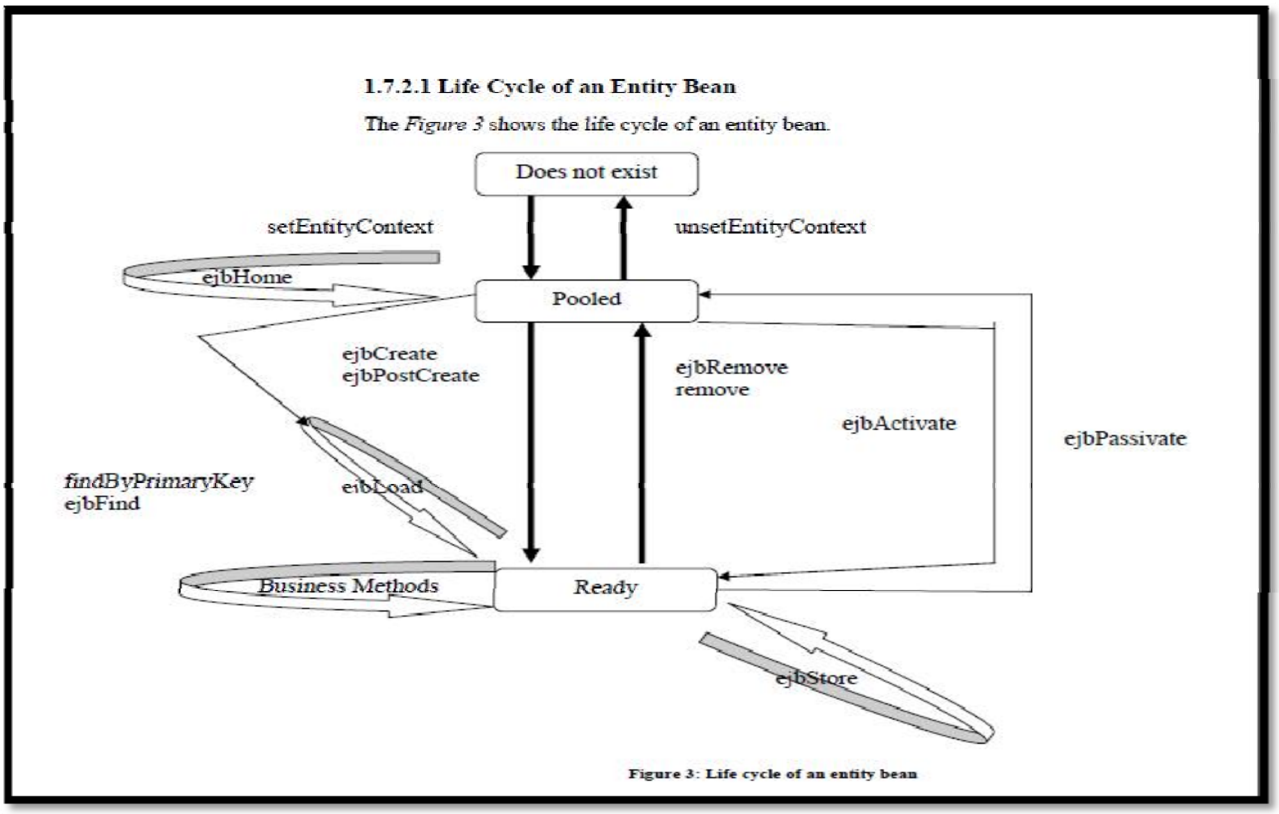
2. Entity Bean

Entity EJBs represent persistent objects. Their lifetimes are not related to the duration of interaction with clients. In nearly all cases, entity EJBs is synchronized with relational databases. This is how persistence is achieved. Entity EJBs are always shared amongst clients. A client cannot get an entity EJB to itself. Thus, entity EJBs are nearly always used as a scheme for mapping relational databases into object-oriented applications.

An important feature of entity EJBs is that they have identity—that is, one can be distinguished from another. This is implemented by assigning a primary key to each instance of the EJB, where 'primary key' has the same meaning as it does for database management. Primary keys that identify EJBs can be of any type, including programmer defined classes.

There are two type of persistence that entity EJB supports. These persistence types are:

- **Bean-managed persistence (BMP):** The entity bean's implementation manages Persistence by coding database access and updating statements in callback methods.
- **Container-managed persistence (CMP):** The container uses specifications made in the deployment descriptor to perform database access and update statements automatically.



1.7.2.1 Life Cycle of an Entity Bean

The *Figure 3* shows the life cycle of an entity bean.

Figure 3: Life cycle of an entity bean

An entity bean has the following three states:

- ✓ **Does not exist:** In this state, the bean instance simply does not exist.
- ✓ **Pooled state:** When the EJB server is first started, several bean instances are created and placed in the pool. A bean instance in the pooled state is not tied to a particular data, that is, it does not correspond to a record in a database table. Additional bean instances can be added to the pool as needed, and a maximum number of instances can be set.
- ✓ **Ready state:** A bean instance in the ready state is tied to a particular data, that is, it represents an instance of an actual business object.
- ✓ **The Use of the Entity Bean**
- ✓ You could probably use an entity bean under the following conditions:
- ✓ The bean represents a business entity and not a procedure. For example, BookInfoBean would be an entity bean, but BookInfoVerifierBean would be a session bean.
- ✓ The bean's state must be persistent. If the bean instance terminates or if the Application Server is shut down; the bean's state still exists in persistent storage (a database).

3. Message Driven Bean

A message-driven bean acts as a consumer of asynchronous messages. It cannot be called for, directly by clients, but is activated by the container when a message arrives. Clients interact with these EJBs by sending messages to the queues or topics to which they are listening. Although a message-driven EJB cannot be called for, directly by clients, it can call other EJBs itself.

- **Life Cycle of a Message Driven Bean**

The *Figure 4* shows the life cycle of a Message Driven Bean:

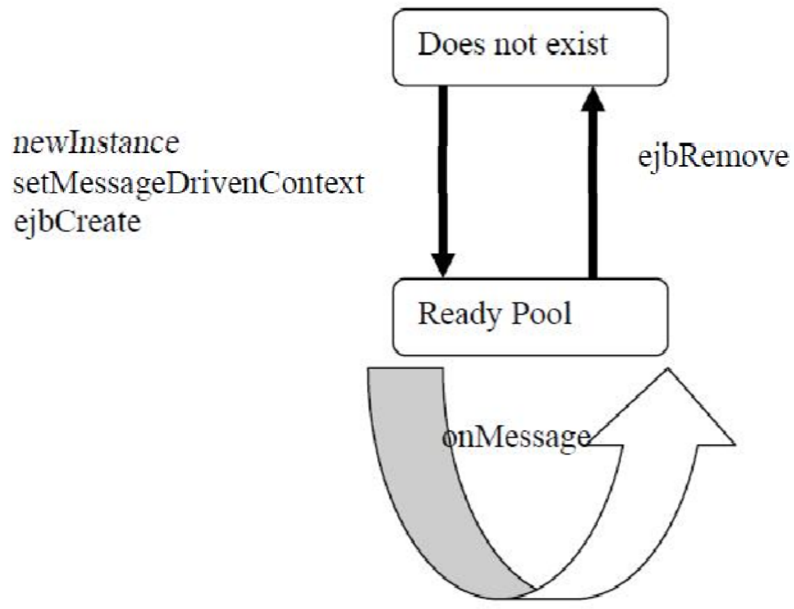


Figure 4: Life Cycle of a Message Driven Bean

A message driven bean has the following two states:

- ✓ **Does not exist:** In this state, the bean instance simply does not exist. Initially, the bean exists in the; does not exist state.
- ✓ **Pooled state:** After invoking the ejbCreate() method, the MDB instance is in the ready pool, waiting to consume incoming messages. Since, MDBs are stateless, all instances of MDBs in the pool are identical; they're allocated to process a message and then return to the pool.
- ✓ **The Use of the Message Driven Bean**

Session beans allow you to send JMS messages and to receive them synchronously, but not asynchronously. To avoid tying up server resources, you may prefer not blocking synchronous receives in a server-side component. To receive messages asynchronously, use a message-driven bean.

III. CONCLUSION

In this research work, we have study the concept of Enterprise JavaBeans. The result of this study is a high-level approach to object-oriented transactional environment for building distributed multi-tier enterprise components. For enterprise application chooses EJB. Based on the requirements either use Session Bean, Entity Bean or Message Driven Bean. Session and Entity beans can be used in normal scenarios.

References

1. Mastering Enterprise JavaBeans Third Edition
2. Beginning EJB 3, Java EE, 7th Edition
3. Mastering Enterprise JavaBeans 3.0, Fourth Edition