EJB 3.0 Notes

Introduction

EJB provides an architecture to develop and deploy component based enterprise applications considering robustness, high scalability, and high performance. An EJB application can be deployed on any of the application server compliant with the J2EE 1.3 standard specification.

We'll be discussing EJB 3.0 in detail in this tutorial.

Types of EJB

EJB is primarily divided into three categories:

1. Session Bean
   Stores data of a particular user for a single session. Can be stateful or stateless.

2. Entity Bean
   Represents persistent data storage. Maps to database records.

3. Message Driven Bean
   Works with JMS. Listens and processes messages asynchronously.

Benefits of EJB

1. Simplified development of enterprise applications.
2. System-level services provided by EJB Container.
3. Automatic lifecycle management of EJB instances.

Stateless Session Bean

A stateless session bean is used for independent operations. No client-specific state is retained between calls.

The container uses a pool of beans to serve client requests.

Steps to Create a Stateless EJB

1. Create a remote/local interface.
2. Use @Local for same-env client; @Remote otherwise.

3. Implement the interface using @Stateless annotation.

Example Code

Remote Interface:

```
@Remote
public interface LibrarySessionBeanRemote {
  // business method declarations
}
```

Stateless EJB:

```
@Stateless
public class LibrarySessionBean implements LibrarySessionBeanRemote {
  // business logic
}
```

Stateful Session Bean Lifecycle

1. Does Not Exist -> Ready: EJB container creates instance.
2. Ready State: Active client interaction.
3. Passivation: Serialized to disk using ejbPassivate().
4. Activation: Restored with ejbActivate().
5. Removal: Client calls remove(), triggers ejbRemove().

Stateless Session Bean Lifecycle

1. Does Not Exist -> Ready: Instances created in pool.
2. Ready State: Used to serve requests.

EJB container manages all transitions automatically.

JNDI (Java Naming and Directory Interface)

JNDI enables discovery and lookup of Java objects via names.

Useful for managing configuration centrally, instead of individual config files.

Properties:

- Initial Factory: Used to communicate with server
- Provider URL: Host:Port for name server
- User & Password: Credentials for JNDI access