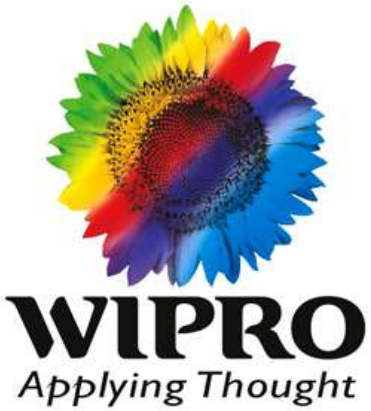


J2EE

EJB 3.0

Agenda

1 **Introduction to EJB 3.0**

2

Objectives

At the end of the module, you will be able to:

- Describe the Architecture of Enterprise Java Beans (EJBs)
- Understand the role of EJBs in Enterprise Applications
- Describe the features of new EJB 3.0 specification

Introduction to EJB 3.0

Enterprise Java Beans

- The Enterprise JavaBeans, which we generally refer as EJB, is a J2EE technology for the development and deployment of component-based business application.
- An enterprise bean is a non-visual component of a distributed, transaction-oriented enterprise application
- Enterprise beans are typically deployed in EJB containers and run on EJB servers
- Applications written using the EJB architecture are:
  - Scalable
  - Transactional
  - Multiuser secure

Enterprise Java Beans

EJB technology defines a server-side, reusable component framework for distributed applications.
It facilitates centrally-managed business logic and declarative deployment. EJB is java based.

Salient features of EJB :

- EJBs are written in Java
- EJBs are executed within an application server environment
- EJBs can be reused
- EJBs can be distributed across many systems and accessed almost as if they were on the local machine
- EJBs are centrally-managed by a container

EJB Runtime Environment

An EJB is essentially a Java object which has no. of interfaces which are exposed to the client and the EJB runs within a container. A container is a runtime environment for EJBs and they cannot exist outside the container. It hosts and manages the beans similar to what a web container does with Servlets.

The EJB container instantiates and controls the enterprise beans and provides them with system-level services.

A client application that wants to access an enterprise bean cannot have direct access to an enterprise bean. The bean is completely isolated from the client application.

Key Features of EJB

EJBs provide:

- A model for specifying server-side components
- A model for providing client interfaces, which are distributed, to the services provided by those components
- Semantics and operations for allowing an EJB container to create, persisit, allocate, activate and destroy instances of the components
- A standard model for creating a components that maintains state information with a client and the session management that is being handled by the EJB container
- A standard model for specifying a component that encapsulates any entry to the data source.

Key Features of EJB

Services provided by an EJB container: A developer, who is responsible for writing classes and interfaces from which an enterprise bean object will be created can assume that the following system-level services will be available from the EJB container:

- Transaction management
- Security
- Persistence
- Remote access
- Lifecycle management
- Database-connection pooling
- Instance pooling

The developer has to concentrate only on developing the business logic for the enterprise bean as the container provides system level services mentioned above.

Types of Enterprise Beans: The EJB architecture defines three types of Enterprise Beans ; **The session bean, The Entity bean and The Message-driven bean**.

Session beans and entity beans are invoked by an enterprise bean client. Message-driven beans, which are usually referred as MDBs, are invoked by a entities, such as a publish/subscribe topic.

Key Features of EJB (Contd.).

A standard model for specifying the configuration characteristics and deployment characteristics of the component, which is again not dependent on the implementation

A standard model for declaratively defining the transactions attributes of a component

A standard component interface contract so that components can run in any vendor-compliant container/server that implements that standard interface contract

However, EJB is not very popular because of the complexity of the EJB architecture.

E.g.: Technologies such as open-source Hibernate framework have overtaken EJB as the preferred choice for developing persistence solutions.

Limitations of EJB 2.1

EJB 2.1 has the following limitations:

- Even for creating a single EJB module, you have to create multiple XML deployment descriptors
- For every bean, you have to create a set of three source files.
- All the callback methods must be implemented which are mostly not used
- Exceptions which are not needed have to be caught
- EJB-QL (EJB-Query Language) is limited in functionality. It is also difficult to use

Limitations of EJB 2.1

- If you are building an EJB with the version 2.x, you have to implement at least two interfaces, the Home Interface and the Remote Interface. You also have to compulsorily build deployment descriptors. Not only that you have to implement EJB specific APIs which increased the complexity level of the application.

- You cannot deploy an 2.x version EJB outside the container. The deployment has to happen on your application server. This is expensive since any change you make to the code will result in redeployment of the appplication.

- In EJB 2.x version applications, you must implement life cycle methods like ejbActive, ejbLoad etc even if they are not needed. Implementing different life cycle methods is mandatory.

Characteristics of EJB

EJB 3.0 is characterized by:

- Use of metadata **annotations**
- **POJO** Programming model
- No compulsion to implement **call back methods.** You can use annotations to mark any method as a callback method to receive life cycle event notifications
- The **"Configuration by Exception**" approach, to simplify the development effort

  The intent is to simplify things for developers by forcing them to code things only where defaults are not adequate
- **Object-Relational Mapping** specified using annotations Based on POJO model
- Encapsulation of **JNDI lookups** using
  - Annotations
  - Dependency injection mechanisms
  - Simple lookup mechanisms

Characteristics of EJB

What's New in EJB 3.0
The main theme of the EJB 3.0 is ease of development, which is the purpose of the new features in this release.

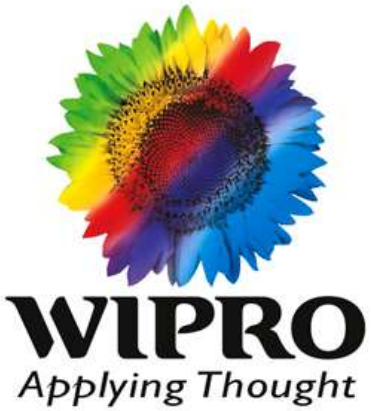
Use of annotations in EJB 3.0 : EJB 3.0 makes use of annotations which can be termed as alternatives to deployment descriptors: In EJB 3.0 version, all EJBs(beans) are Plain Old Java Objects (POJO), with proper annotations. So a developer marks up his/her Java code with annotations and the annotation processor generates the deployment descriptors at the time of execution(runtime). This mechanism allows the deployer to override the default configurations so they can replace data sources, etc. Another advantage over here is that both the code and annotations are in a single file and the developer doesn't have to maintain multiple files for one bean(as in the case of EJB 2.x).

Callback Methods and Listener Classes: In EJB 3.0, bean developers do not have to mandatorily implement unnecessary callback methods. They can create any method and annotate this method as a callback method to receive notifications for lifecycle events whether it is for a SessionBean or a MessageDrivenBean (MDB).

Summary

In this module, you were able to:

- Describe the Architecture of Enterprise Java Beans (EJBs)
- Understand the role of EJBs in Enterprise Applications
- Describe the features of new EJB 3.0 specification
- Differentiate between EJB 2.1 and EJB 3.0 specifications



Thank You