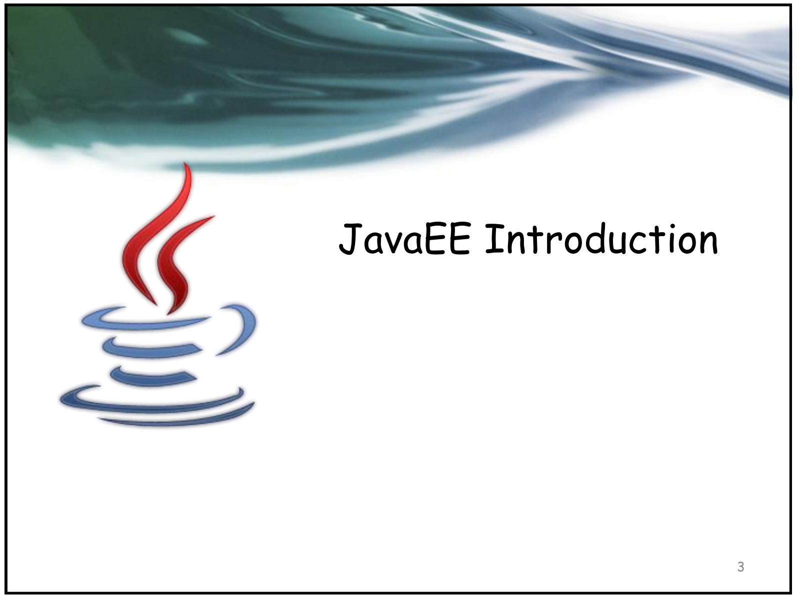
JavaEE 7 Introduction

By Võ Văn Hải
Faculty of Information Technologies
Industrial University of Ho Chi Minh City

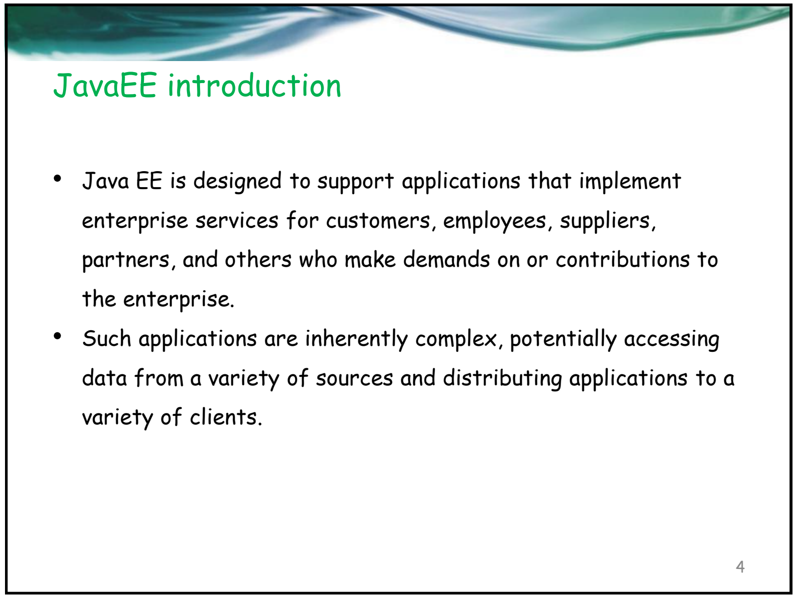
Session objectives

- JavaEE 7 Introduction
- Enterprise Beans
- Persistence

2



JavaEE Introduction



JavaEE introduction

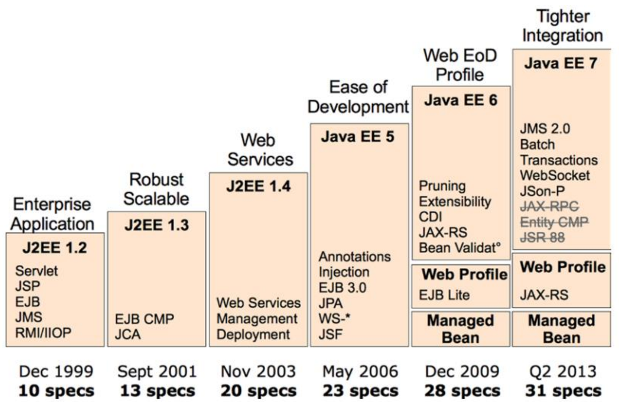
- Java EE is designed to support applications that implement enterprise services for customers, employees, suppliers, partners, and others who make demands on or contributions to the enterprise.
- Such applications are inherently complex, potentially accessing data from a variety of sources and distributing applications to a variety of clients.

JavaEE introduction

- The Java EE application model defines an architecture for implementing services as multitier applications that deliver the scalability, accessibility, and manageability needed by enterprise-level applications. This model partitions the work needed to implement a multitier service into the following parts:
  - The business and presentation logic to be implemented by the developer
  - The standard system services provided by the Java EE platform

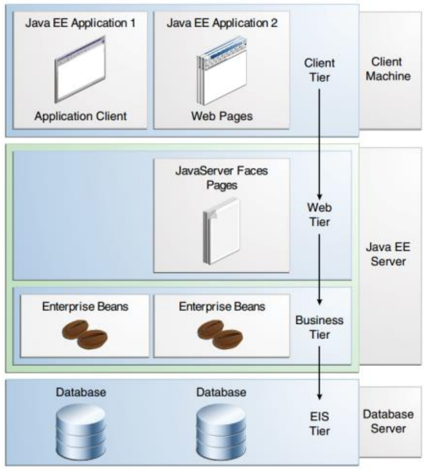
5

History of J2EE/Java EE



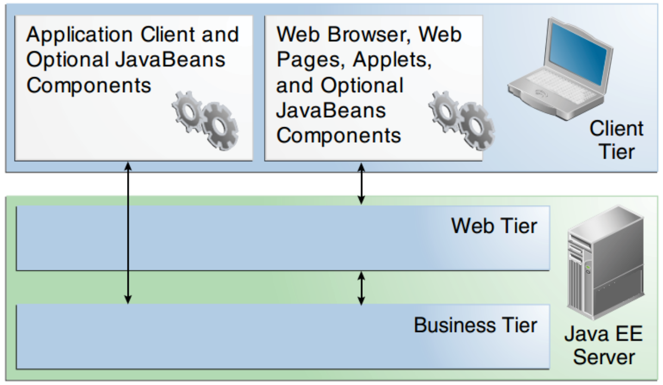
6

Distributed Multi-tiered Applications



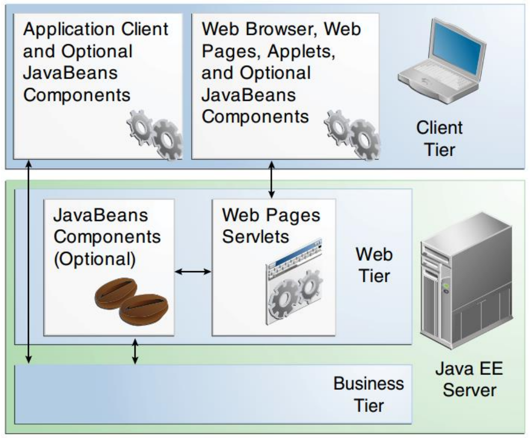
8

Server communication


Application Client and Optional JavaBeans Components
Web Browser, Web Pages, Applets, and Optional JavaBeans Components
Client Tier
Web Tier
Business Tier
Java EE Server

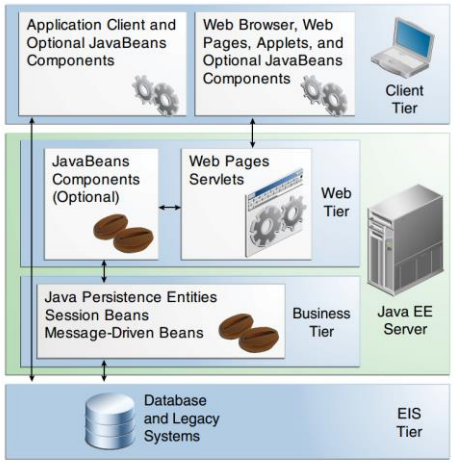
9

Web Tier and Java EE Applications


Application Client and Optional JavaBeans Components
Web Browser, Web Pages, Applets, and Optional JavaBeans Components
Client Tier
JavaBeans Components (Optional)
Web Pages Servlets
Web Tier
Business Tier
Java EE Server

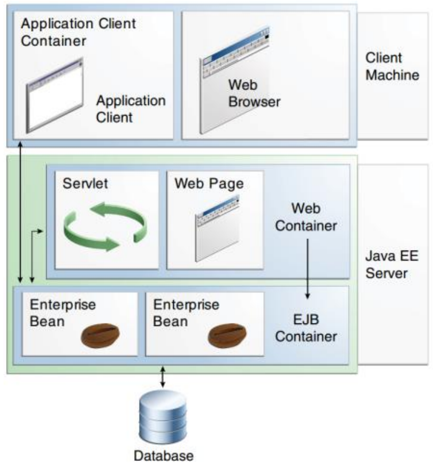
10

Business and EIS Tiers

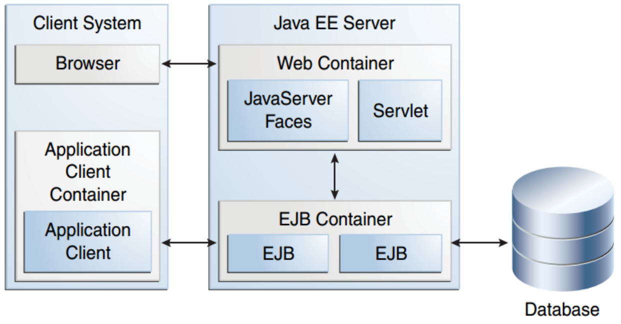

Application Client and Optional JavaBeans Components
Web Browser, Web Pages, Applets, and Optional JavaBeans Components
Client Tier
JavaBeans Components (Optional)
Web Pages Servlets
Web Tier
Java Persistence Entities Session Beans Message-Driven Beans
Business Tier
Java EE Server
Database and Legacy Systems
EIS Tier

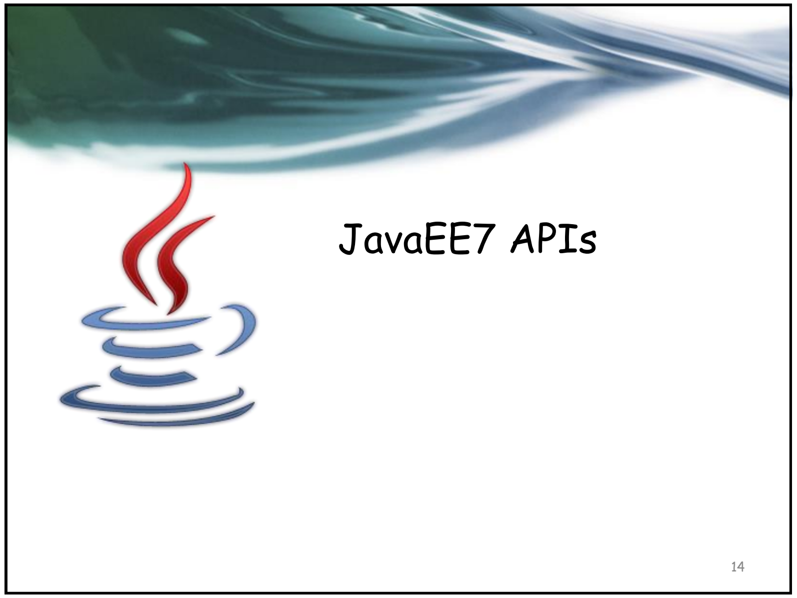
11

Java EE Server and Containers

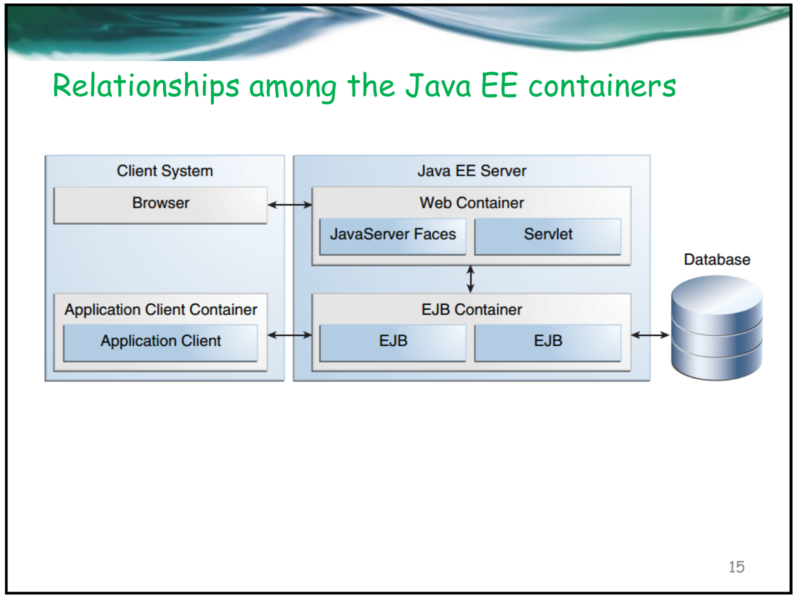


Java EE Containers

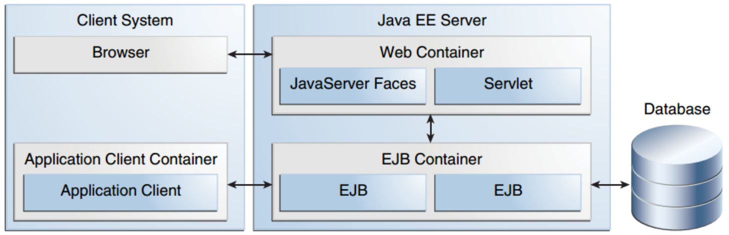




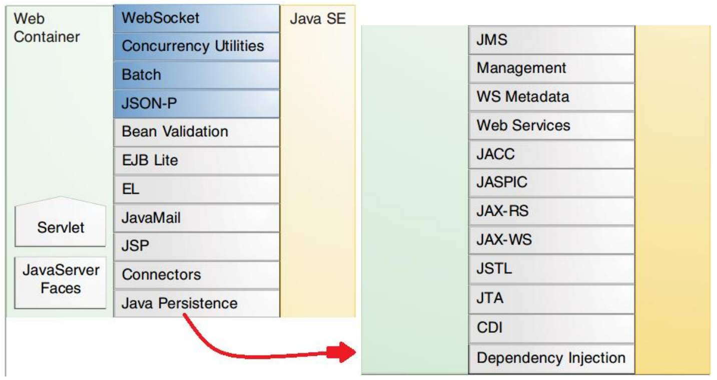
JavaEE7 APIs



Relationships among the Java EE containers

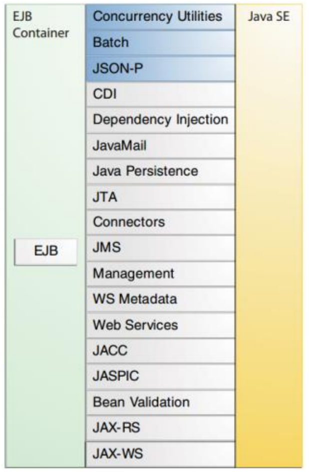


Java EE APIs in the Web Container



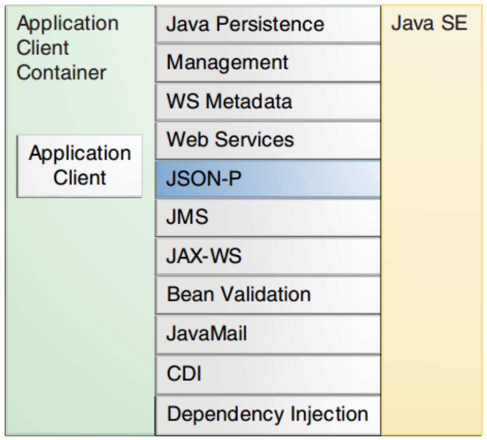
16

Java EE APIs in the EJB Container



17

Java EE APIs in the Application Client Container



18

Enterprise JavaBeans Technology

- An Enterprise JavaBeans (EJB) component, or enterprise bean, is a body of code having fields and methods to implement modules of business logic.
- You can think of an enterprise bean as a building block that can be used alone or with other enterprise beans to execute business logic on the Java EE server.
- Enterprise beans are either session beans or message-driven beans.
  - A **session bean** represents a transient conversation with a client. When the client finishes executing, the session bean and its data are gone.
  - A **message-driven bean** combines features of a session bean and a message listener, allowing a business component to receive messages asynchronously. Commonly, these are Java Message Service (JMS) messages.

19

Java Persistence API

- The Java Persistence API (JPA) is a Java standards-based solution for persistence.
- Persistence uses an object/relational mapping approach to bridge the gap between an object-oriented model and a relational database.
- The Java Persistence API can also be used in Java SE applications, outside of the Java EE environment.
- Java Persistence consists of the following areas:
  - The Java Persistence API
  - The query language
  - Object/relational mapping metadata

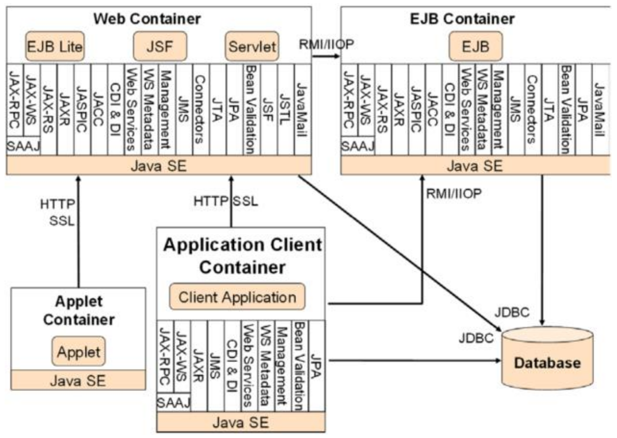
20

Other Technologies

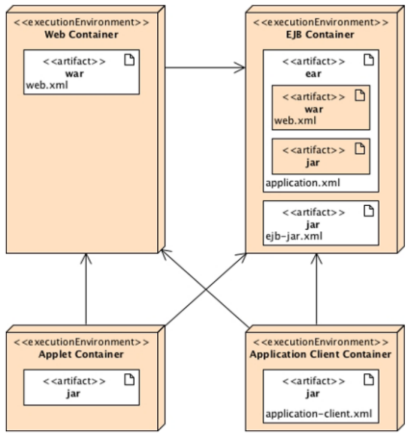
- Java Servlet Technology
- JavaServer Faces Technology
- JavaServer Pages Technology
- Java Transaction API
- …

21

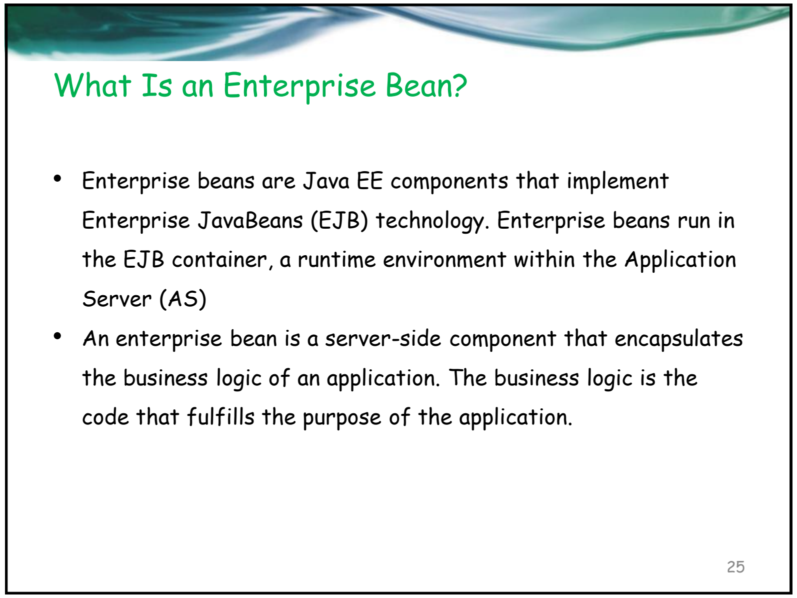
Services provided by containers



Packaging - Archives in containers



Enterprise Beans



What Is an Enterprise Bean?

- Enterprise beans are Java EE components that implement Enterprise JavaBeans (EJB) technology. Enterprise beans run in the EJB container, a runtime environment within the Application Server (AS)
- An enterprise bean is a server-side component that encapsulates the business logic of an application. The business logic is the code that fulfills the purpose of the application.

When to Use Enterprise Beans

- You should consider using enterprise beans if your application has any of the following requirements:
  - The application must be scalable. To accommodate a growing number of users, you may need to distribute an application's components across multiple machines. Not only can the enterprise beans of an application run on different machines, but also their location will remain transparent to the clients.
  - 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.

26

Types of Enterprise Beans

Enterprise Bean Type	Purpose
Session	Performs a task for a 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

27

What Is a Session Bean?

- A session bean encapsulates business logic that can be invoked programmatically by a client over local, remote, or web service client views.
- To access an application that is deployed on the server, the client invokes the session bean's methods.
- The session bean performs work for its client, shielding it from complexity by executing business tasks inside the server.
- A session bean is not persistent. (That is, its data is not saved to a database.)

Types of Session Beans

- Session beans are of three types:
  - stateful,
  - stateless,
  - and singleton.

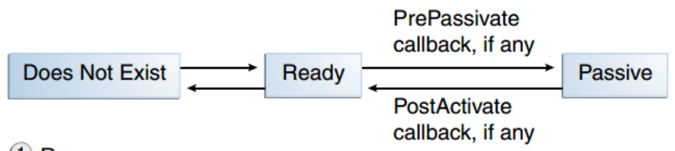
Types of Session Beans
Stateful Session Beans

- The state of an object consists of the values of its instance variables.
- In a stateful session bean, the instance variables represent the state of a unique client/bean session.
- Because the client interacts ("talks") with its bean, this state is often called the **conversational state**.
- A session bean is not shared; it can have only one client, in the same way that an interactive session can have only one user. When the client terminates, its session bean appears to terminate and is no longer associated with the client.
- The state is retained for the duration of the client/bean session. If the client removes the bean, the session ends and the state disappears.

30

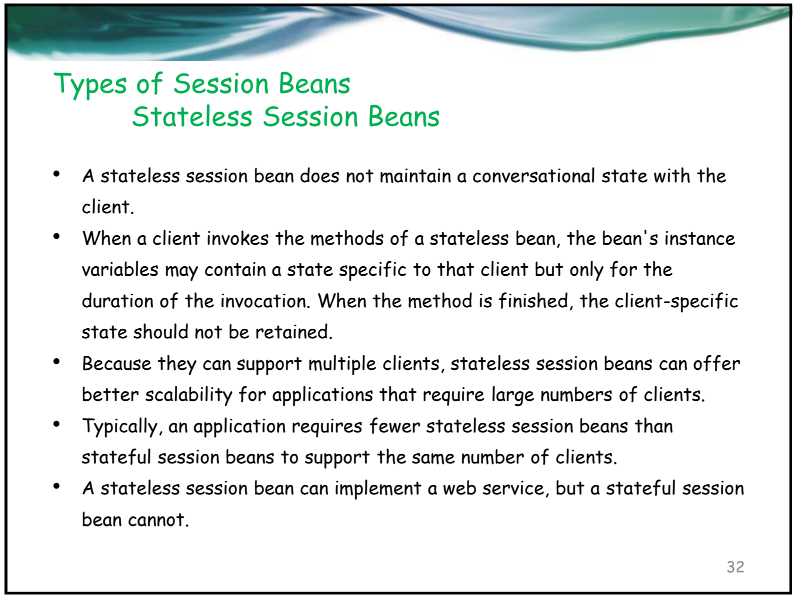
Lifecycle of a Stateful Session Bean

① Create
② Dependency injection, if any
③ PostConstruct callback, if any
④ Init method, or ejbCreate<METHOD>, if any



① Remove
② PreDestroy callback, if any

31

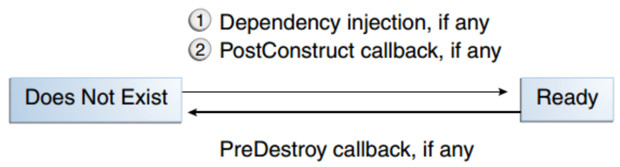


Types of Session Beans
Stateless Session Beans

- A stateless session bean does not maintain a conversational state with the client.
- When a client invokes the methods of a stateless bean, the bean's instance variables may contain a state specific to that client but only for the duration of the invocation. When the method is finished, the client-specific state should not be retained.
- Because they can support multiple clients, stateless session beans can offer better scalability for applications that require large numbers of clients.
- Typically, an application requires fewer stateless session beans than stateful session beans to support the same number of clients.
- A stateless session bean can implement a web service, but a stateful session bean cannot.

32

The Lifecycle of a Stateless Session Bean

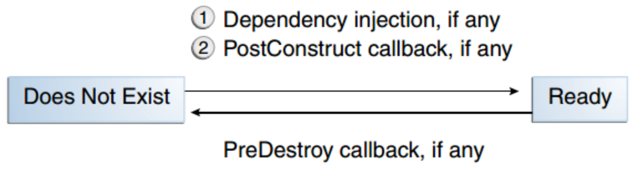

① Dependency injection, if any
② PostConstruct callback, if any
Does Not Exist
Ready
PreDestroy callback, if any

33

Types of Session Beans
Singleton Session Beans

- A singleton session bean is instantiated once per application and exists for the lifecycle of the application. Singleton session beans are designed for circumstances in which a single enterprise bean instance is shared across and concurrently accessed by clients.
- Singleton session beans maintain their state between client invocations but are not required to maintain their state across server crashes or shutdowns.

The Lifecycle of a Singleton Session Bean

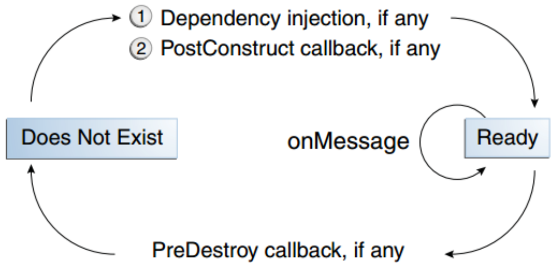

① Dependency injection, if any
② PostConstruct callback, if any
Does Not Exist
Ready
PreDestroy callback, if any

What Is a Message-Driven Bean?

- A message-driven bean is an enterprise bean that allows 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.
- The messages can be sent by any Java EE component (an application client, another enterprise bean, or a web component)or by a JMS application or system that does not use Java EE technology.
- Message-driven beans can process JMS messages or other kinds of messages.

36

The Lifecycle of a Message-Driven Bean


① Dependency injection, if any
② PostConstruct callback, if any
Does Not Exist
onMessage
Ready
PreDestroy callback, if any

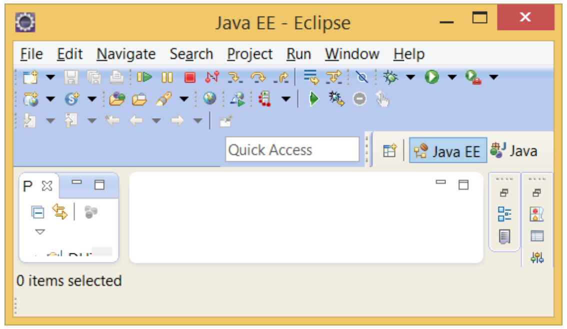
37

Comparison Between EJB Lite and Full EJB

Feature	EJB Lite	Full EJB 3.2
Session beans (stateless, stateful, singleton)	Yes	Yes
No-interface view	Yes	Yes
Local interface	Yes	Yes
Interceptors	Yes	Yes
Transaction support	Yes	Yes
Security	Yes	Yes
Embeddable API	Yes	Yes
Asynchronous calls	No	Yes
MDBs	No	Yes
Remote interface	No	Yes
JAX-WS web services	No	Yes
JAX-RS web services	No	Yes
Timer service	No	Yes
RMI/IIOP interoperability	No	Yes

38

Live Demo



39

Persistence

Introduction to the Java Persistence API

- The Java Persistence API provides Java developers with an object/relational mapping facility for managing relational data in Java applications. Java Persistence consists of four areas:
  - The Java Persistence API
  - The query language
  - The Java Persistence Criteria API
  - Object/relational mapping metadata

View slide

Read more…

http://docs.oracle.com/javaee/7/tutorial/doc/javaeetutorial7.pdf

42