Session 7:

Enterprise Java Bean (EJB)

Contents

- JavaBeans and Enterprise JavaBeans
- EJB Components
- Three Types of EJB
  - Session Bean
  - Entity Bean
  - Message Bean

Defining Enterprise JavaBeans(1)

- A server-side component architecture
- Model to enable efficient development and deployment of Java applications :
  - Transactional, Portable
  - Distributed, Multi-tier
  - Scalable
  - Secure

Defining Enterprise JavaBeans (cont.)

- EJB is not JavaBeans
- a Server Component specification for Java
- Separates business and system programming
- Portability of business objects
- Extensibility through vendor features

Defining JavaBeans(1)

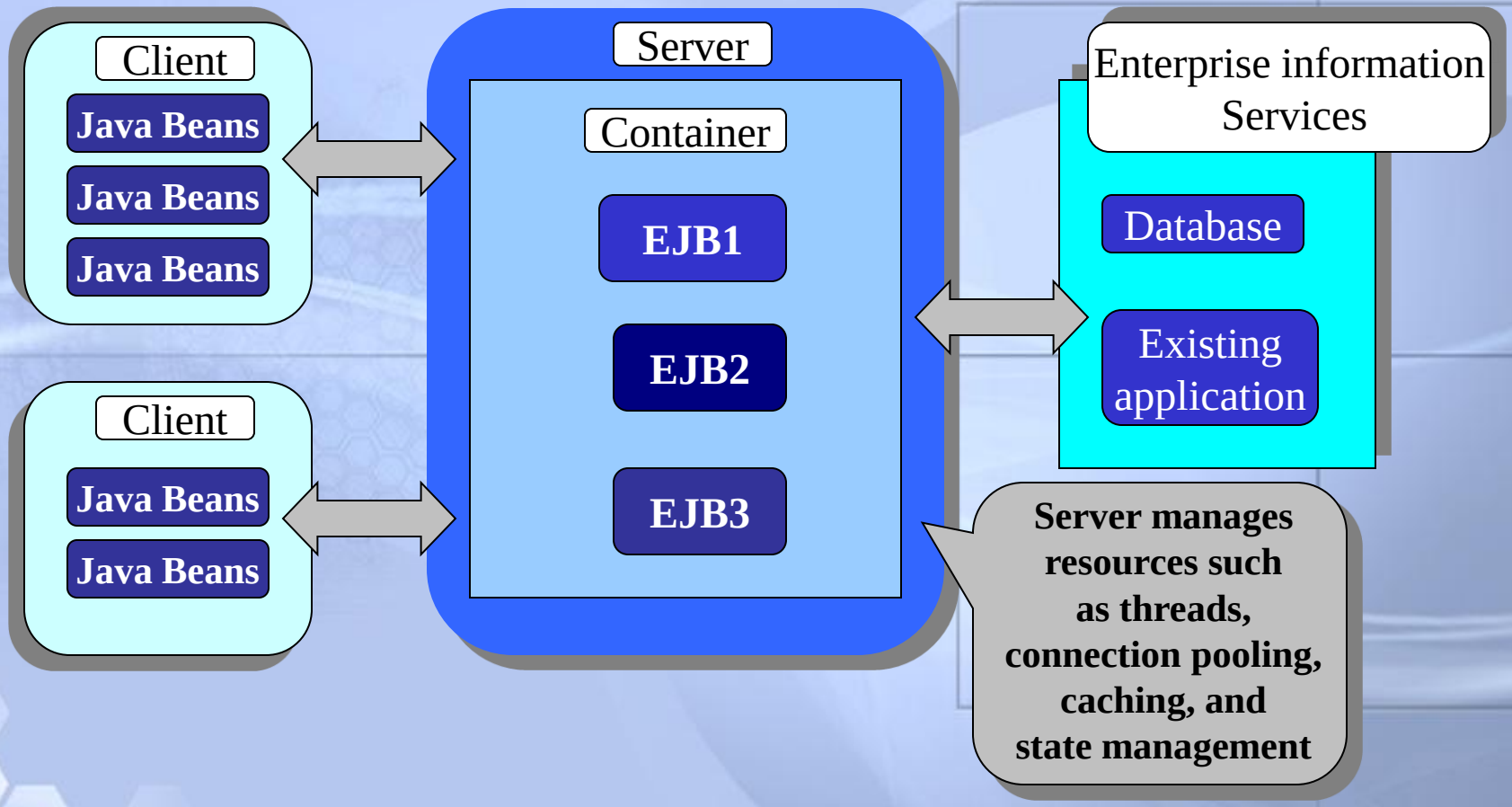
- A client-side component architecture
- Portable, platform-independent component model written in the java programming language
- Acts as a bridge between proprietary component models
- Provides a seamless and powerful means for developers to build components that run in ActiveX container applications

JavaBeans Vs. Enterprise JavaBeans(1)

- JavaBeans
  - Can be either visible or non-visible
  - Client > Server
  - java.bean.*
  - Intra-processor
  - Easier to develop than EJB

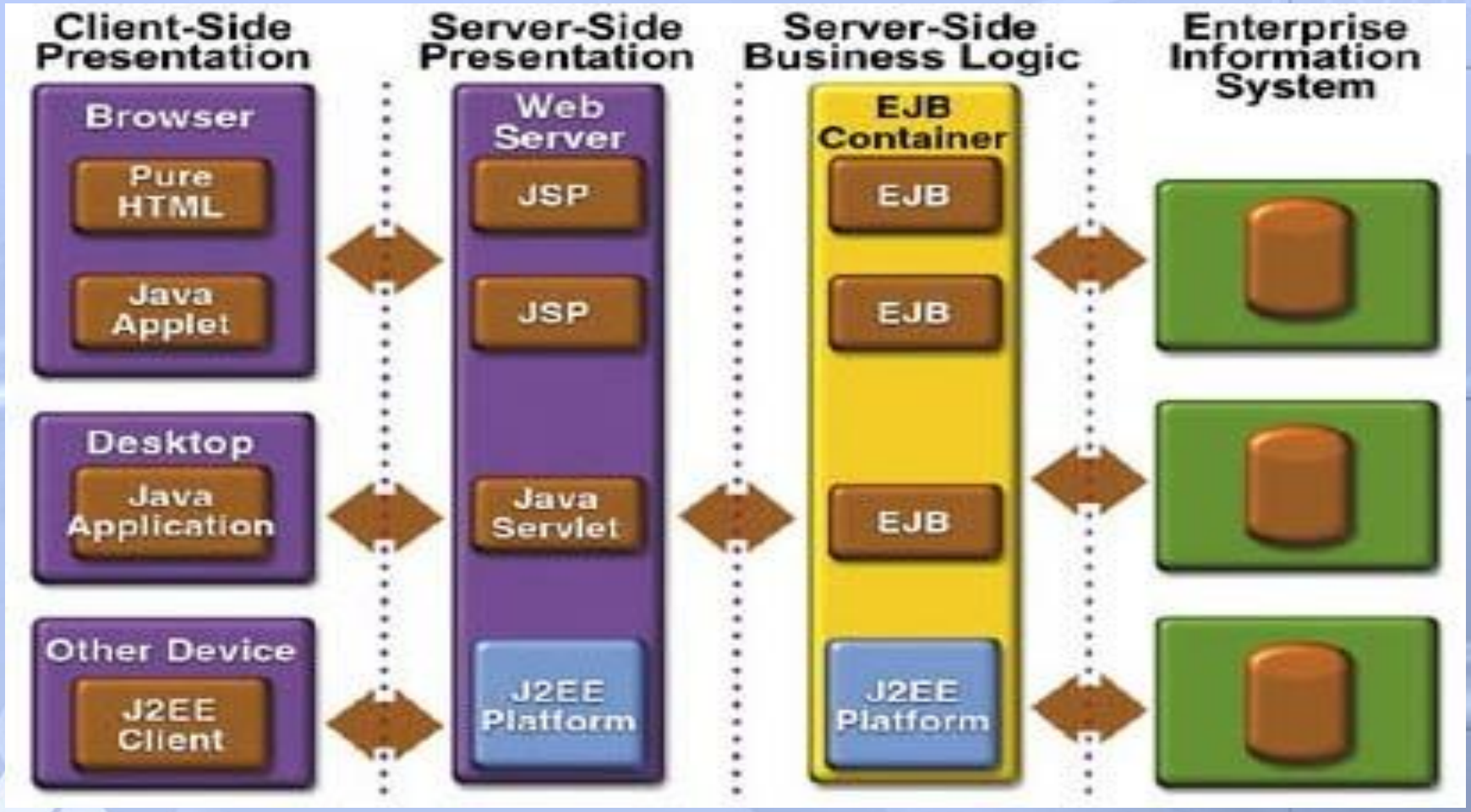
- EJB
  - Are decidedly non-visible, remote objects
  - Server
  - javax.ejb.*
  - Inter-processor
  - More difficult to develop than JavaBeans

JavaBeans Vs. Enterprise JavaBeans(2)

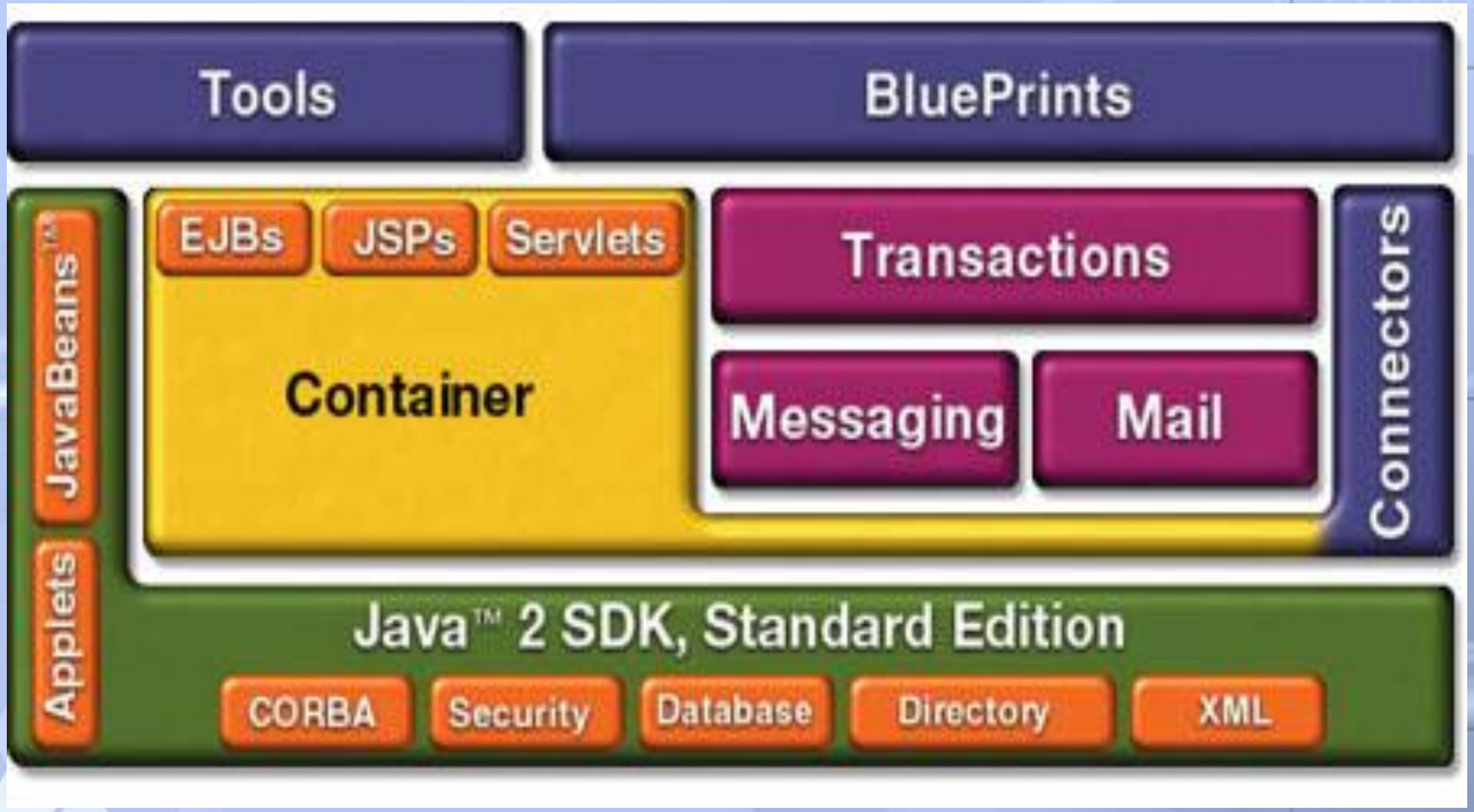


7

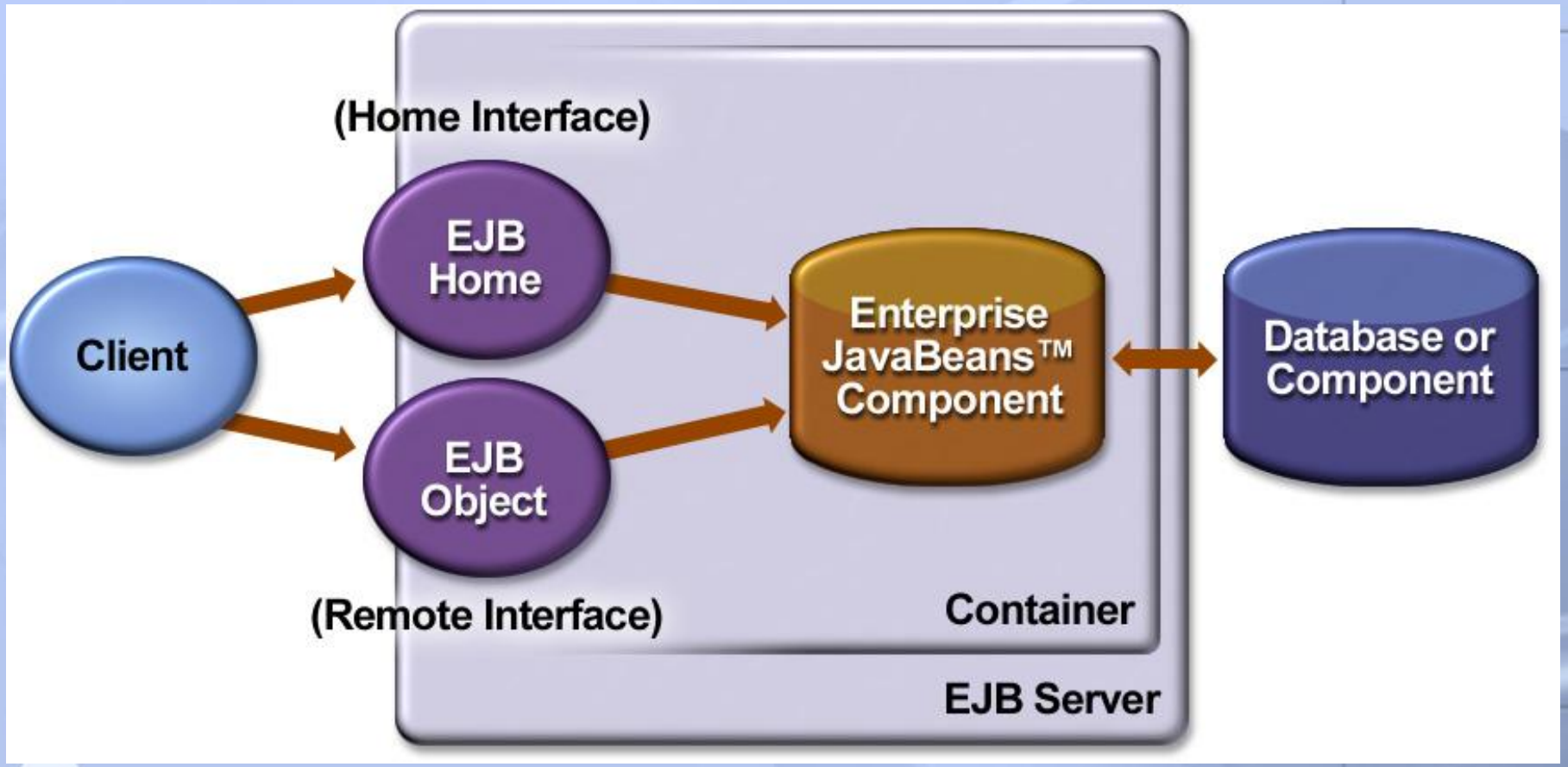
Where they fit in a system



EJB's in J2EE


Tools
BluePrints
JavaBeans™
Applets
EJBs
JSPs
Servlets
Container
Transactions
Messaging
Mail
Connectors
Java™ 2 SDK, Standard Edition
CORBA
Security
Database
Directory
XML

EJB Architecture



Components of EJB Architecture

- **EJB Server**
- **EJB Client**
- **EJB Container**

❖ EJB Container runs on **EJB Server**.

❖ EJBs run in **EJB Container**.

❖ **EJB Clients** use EJBs through **Interfaces**.

EJB Server

- Provides the system services like
  - Multiprocessing
  - Load-balancing
  - Device access
  - A raw execution environment
- Provides Naming and Transaction Services

EJB Client

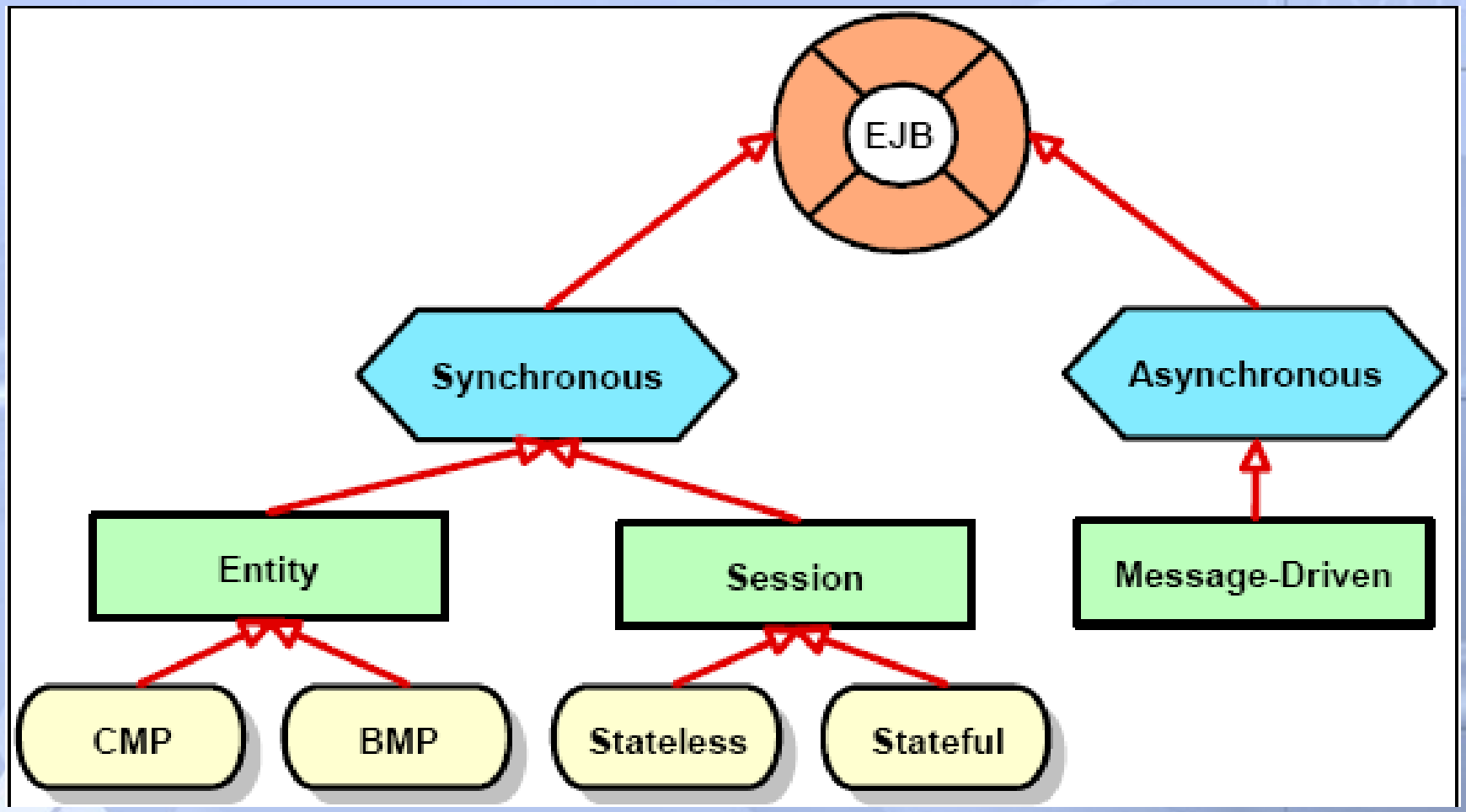
- They locate the EJB Container that contains the bean through the Java Naming and Directory (JNDI) Interface.

- They use EJB Container to invoke EJB bean methods.

EJB Container

- **Interface provider** between an Enterprise JavaBean and outside world.
- An EJB Client has no access to a bean directly.
- Any bean access is done through Container generated methods which can turn invoke the bean methods.
- Two types of Containers:
  1. Section Container
     - Contains transient, non-persistent EJBs
  2. Entity Container
     - Contains persistent EJBs

EJB components



Three kinds of EJB's

- **Session**
  - associate client information with a specific client
  - both stateless and stateful versions
- **Entity**
  - groups associated information in an abstraction that provides transaction support
- **Message Bean**
  - rarely used, hardly supported

What is a Session Bean?

- Represents a single Client inside the J2EE server
- One client at a time/ not persistent
- when the client terminates, the session bean is disassociated from the client
- There are two types: **Stateful and Stateless**

Stateful

- These represent a set of interactions between client and server.
  - Example: shopping cart
- Saves information over several method invocations.
- There is a lot of overhead associated with using stateful beans

Stateless Beans

- A stateless bean does not save information between method calls.
- Limited application
- Little overhead
  - multiple clients can use the same bean instance without alteration
- Example: fetch from a read-only database or send a confirmation email for an order

Entity Beans

- Associates pieces of information in a group
- Accessed by multiple clients at a time
- Persistent and Serializable
- The container loads and stores the entity beans in the database
- These are more similar to regular beans

More on Entity Beans

- **Transactions**: this is what makes an Entity Bean special.
  - Entity beans rely on the container to enforce robust transactions
  - example:

    Airline booking: if the flight booking action fails, then the credit card charge action fails, or vice versa.

Persistence in Entity Beans

- **Container managed persistence**
  - the container controls when the bean is read from or written to the database
- **Bean managed persistence**
  - the bean's implementation performs all of the sql operations that loads, stores, and updates the bean's data to or from the database.
  - Bean is responsible for connection allocation to the database

EJB Interfaces

- **Remote Interface**
  - Business end of EJB
  - The set of actual services provided by the EJB
- **Home Interface**
  - Book-keeping interface
  - Helps the client to create a new instance of an EJB or to find the existing interface of the EJB

Benefits of Enterprise Beans

- The developer can solve business problems in ease.
- Because the beans contain the application's business logic, the client developer can concentrate on the presentation of the client.
- The clients are thinner.
- The application becomes the portable components.

Limitations of Enterprise Beans

- Enterprise beans are restricted from performing certain operations as follows:
  - Managing or synchronizing threads
  - Accessing files or directories with the java.io package
  - Using AWT functionality
  - Listening on a socket, accepting connections on a socket
  - Loading a native library

Database Access

- Both the Session bean and Entity beans can access a database.
- **Done in Session bean when**:
  - The application is relatively simple
  - The data returned by the SQL call will not be used by multiple clients
  - The data does not represent a business entity

- **Done in Entity bean when:**
  - Multiple clients will use the data returned by the database call
  - The data represent a business entity
  - You desire to hide the relational model from the Session bean

Connection Pooling

- Setting up connections to the database is resource intensive
- Connection pooling maintains a pool of database connections for the entity beans so that the connection is maintained when a bean finishes, and is available for other entity beans.
- Specific to database and EJB container implementation

Using an Entity bean from a Session bean

- An entity bean can be shared by multiple sessions.
  - This allows for data encapsulation; clients can interact with data via session beans within transaction boundaries.

- Can do all database interaction from session bean as an alternative
  - encapsulation is weakened

Underlying Technologies RMI

- **RMI - Remote Method Invocation**
  - instead of invoking a method on another Java object running in the same JVM, you invoke a method in a Java object in another JVM on the same computer or another one.
  - Using an interface, RMI hides the fact that you are invoking a method remotely.
  - The Remote and Home interfaces for an EJB must be RMI interfaces.

Underlying Technologies: JNDI

- **JNDI - Java Naming and Directory Interface**
  - JNDI provides a uniform way to access naming and directory services.
  - You use JNDI to locate EJB's and JDBC connection pools from within your EJB container.
  - When a client needs to access a bean's Home interface, it uses JNDI to locate the Home interface. After you locate an object, you communicate directly with it instead of going through JNDI
  - You don't need to know much about JNDI for EJB's except for a few setup calls.

Underlying Technologies: JDBC

- **JDBC - Java Database Connectivity**
  - gives you a standard API in which to communicate with different types of databases
  - If you use CMP (Container Managed Persistence) there's a chance that you won't use JDBC at all. However there are still a few cases in which CMP doesn't handle all the different ways that you can access data.

Thank You!