CmpE 473
Internet Programming

Pınar Yolum
pinar.yolum@boun.edu.tr

Department of
Computer Engineering
Boğaziçi University

Enterprise Java Beans

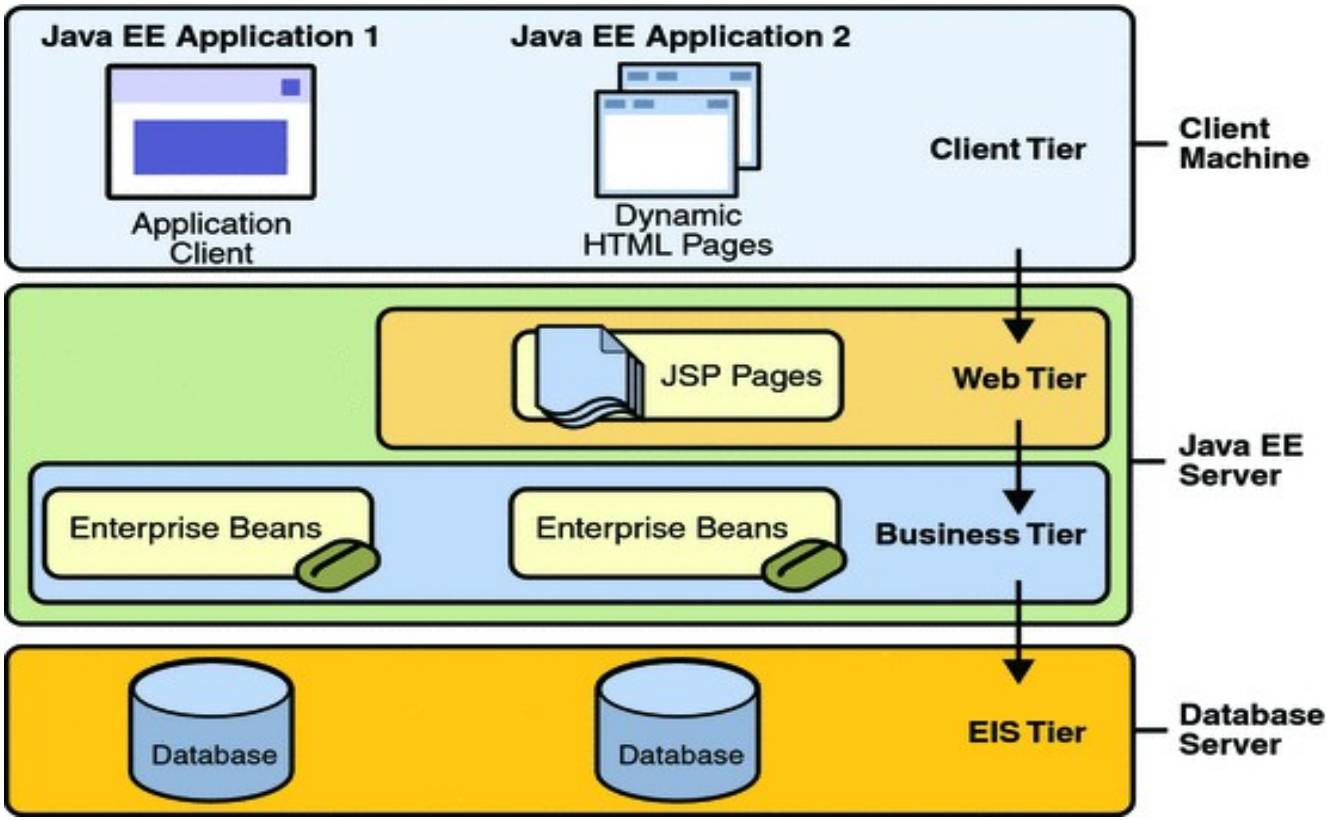
Two-Tier Architecture

- Client tier
  - Access the server to get a request fulfilled
- Server tier
  - Fulfils business requests
  - User interface + Database + Business logic
  - No layers; no abstraction

Three-Tier Architecture

- Three separate layers
- Presentation tier
  - Runs on client
  - Provides user interface
  - Invokes business logic
- Business logic tier
  - Runs on server
  - Has application logic, business rules, etc.
- Data tier
  - Runs on a database server
  - Stores and provides access to data
- Advantages?

Distributed Multitier Application

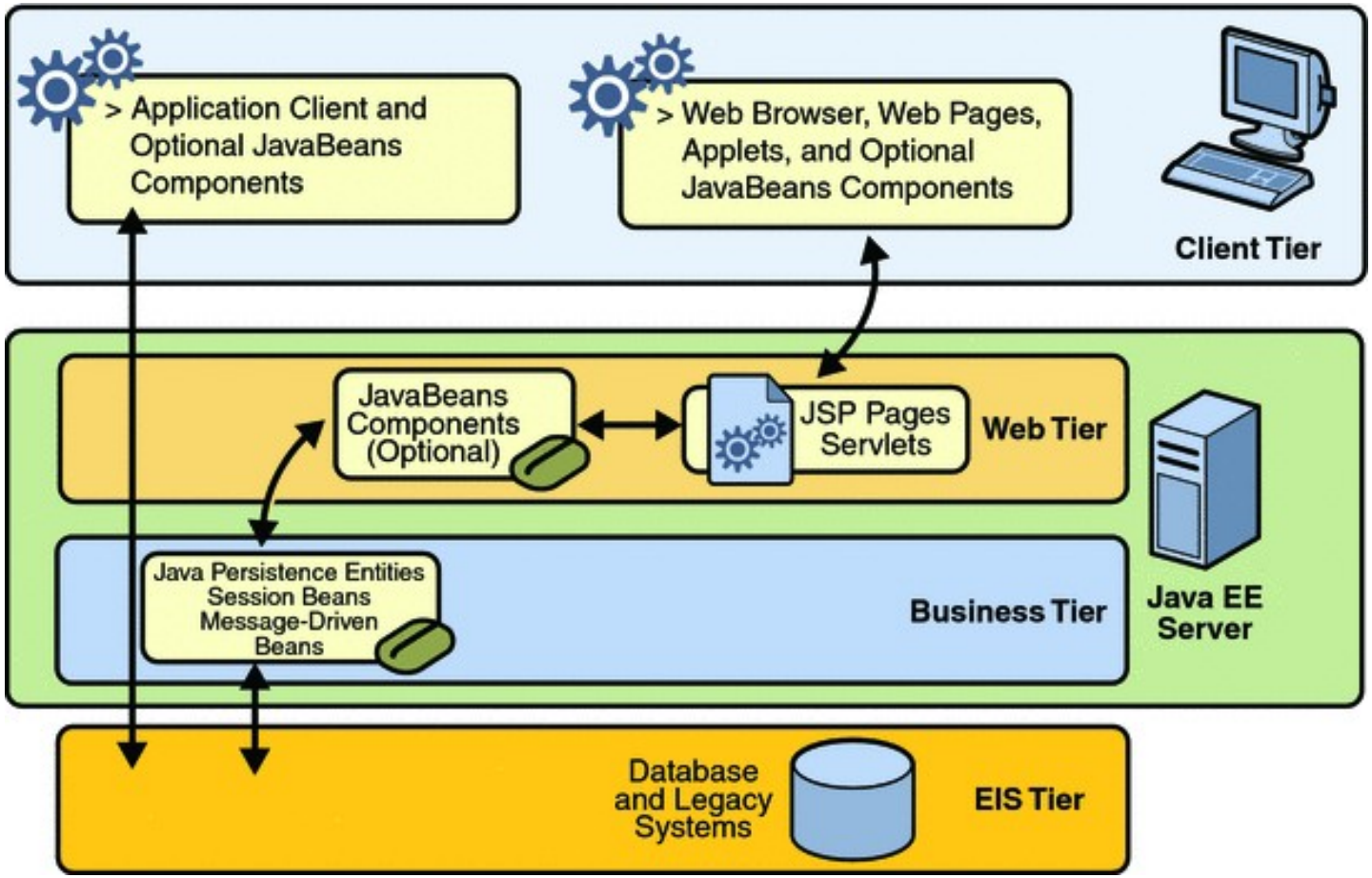


Diagrams from EE Tutorial

Multi-Tier Architecture

- Client-tier components run on the client machine.
- Web-tier components run on the J2EE server.
- Business-tier *components* run on the J2EE server.
  - Divide the application logic into multiple components
  - Components may exist in different machines
- Enterprise information system (EIS)-tier software runs on the EIS server.

J2EE-Based Design

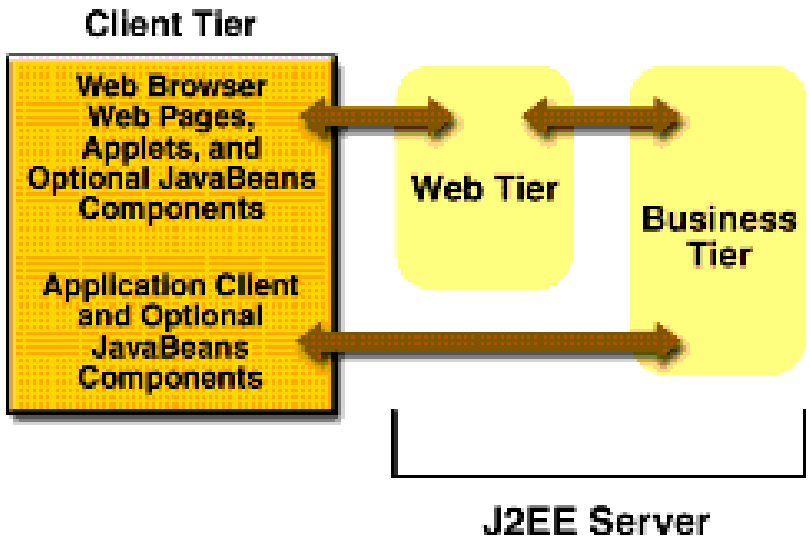


Component

- Functional component
  - Self-contained (contains related class files)
  - Exists in an application (J2EE application)
  - Communicates with other components
- J2EE Components
  - Client components: Application clients and applets
  - Server side Web components: Servlet and JavaServer Pages (JSP)
  - Business components: Enterprise JavaBeans

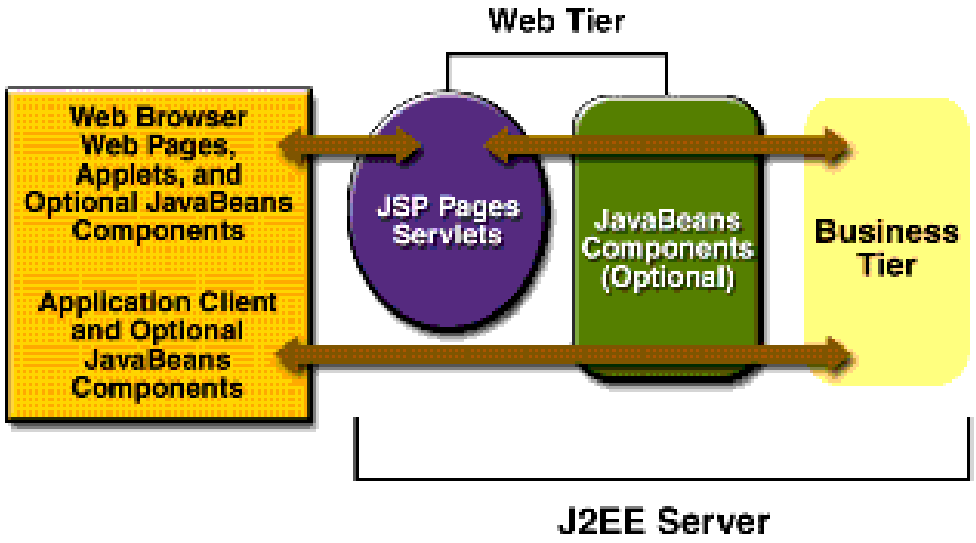
Server Communications

- Thin client
  - Browser-based
  - Move most functionality to server
  - Easy to distribute and manage application (such as updates)
- Thick client
  - Stand alone application
  - More functionality
  - Better user experience

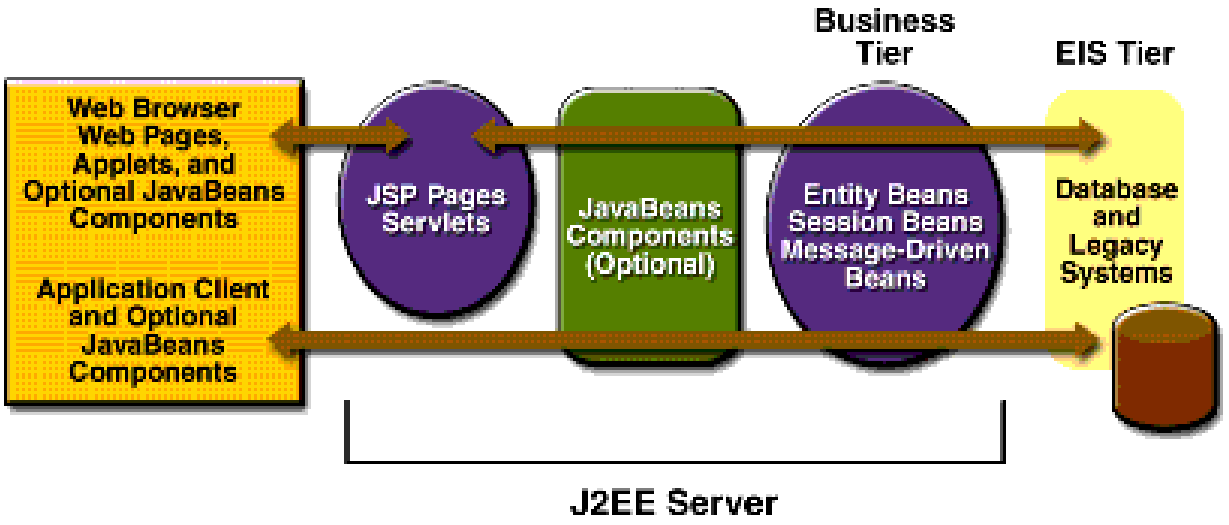


Web components

- Servlet
  - Java classes
  - Process requests dynamically
- JavaServer Pages
  - Text-based but execute like servlets
  - Include static content easily



Business components

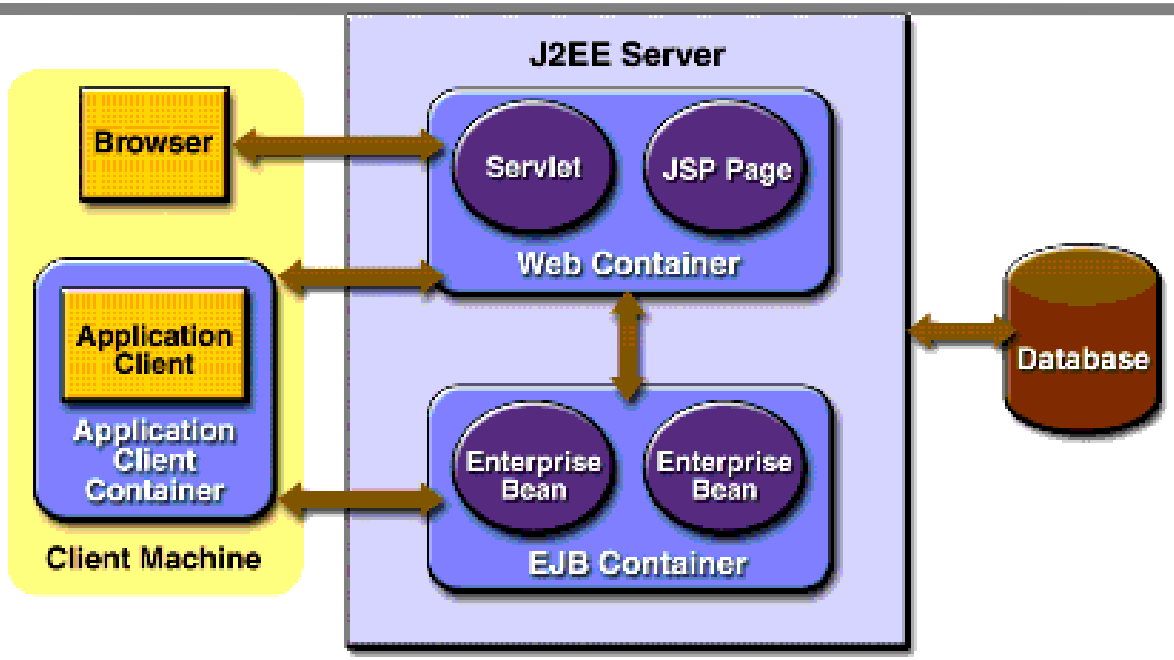


- Enterprise beans
  - Business logic: banking, course registration
  - Receive data from clients, process, and store in EIS and vice versa
  - EIS: Database system, ERP

Container

- Division of work
  - Many details related to underlying architecture
  - Operating system details
  - Communication with system resources
  - Focus on business logic
- Container
  - Support components
  - Provide interface to system functionality
  - No need to know the underlying details
- Assemble all components into a J2EE module
- Deploy them in containers

Container



- Customize container support for components
  - Configure a web component to give access to authorized users only
- Non-configurable services
  - Data persistence
  - Servlet life-cycle
- Containers depend on the tiers

Design of Business Logic

- Components for each functionality
  - May run on different servers
  - Location may need to be transparent
- Should support transactions
  - Ensure data integrity
  - Concurrent access to data
- Must be independent of client architecture
  - Thin clients over browsers
  - Thick clients
- Must be scalable
  - Serve many clients at the same time

Enterprise Beans

- Session beans
  - Transient interactions with client
  - Data gone after the execution
- Message-driven bean
  - Session bean + JMS message listener
  - Allow business components to receive messages (asynchronously)
- Entity bean (Replaced with Java Persistence API)
  - Persistent interactions with client
  - Data stored after the client exits

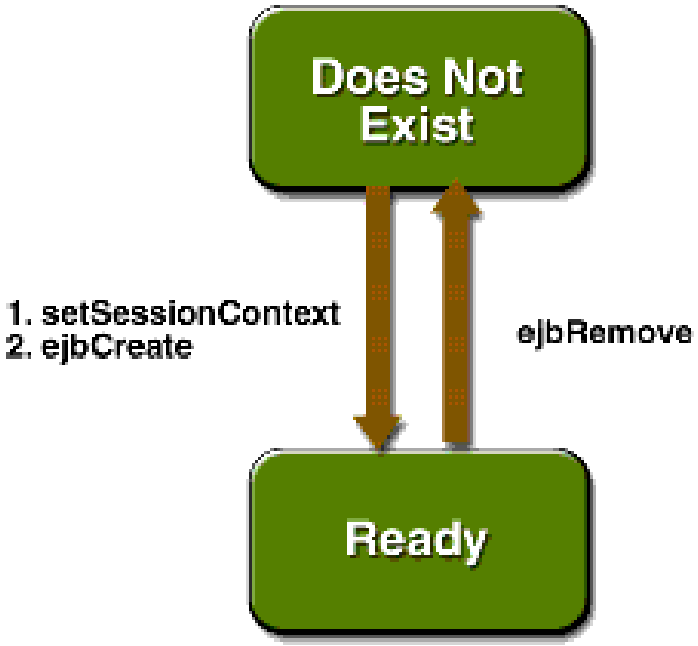
Session Beans

- Interactive session with a client
- Serves one client at a time
- Stateless (no conversational state)
  - During the invocation variables get values
  - Dropped after invocation
  - Any instance can be assigned to a client
  - Never stored in secondary storage
- Stateful
  - Holds state between method invocations
  - The values of variable represent unique state

Session Beans

- When to capture state?
  - If the bean is an intermediary between the client and other components
  - If the bean coordinates the activities of multiple enterprise beans
- When not to capture state?
  - If the bean data is not specific to client
  - No relation between method invocations
  - The bean delivers the same information to clients

Session Beans

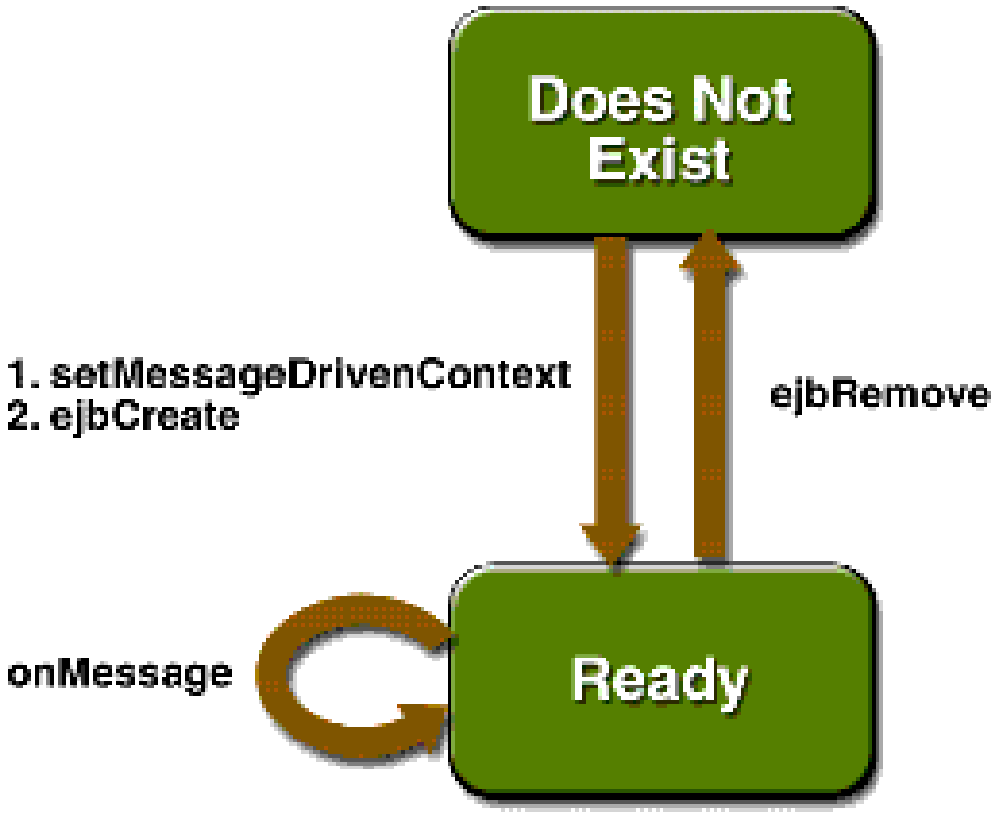


- Stateful:
  - Client invokes create method and EJB container initiates the bean
  - Passivate: Move to secondary storage (LRU)
  - Create and remove: Handled by application programmer

Message-Driven Beans

- Message listeners for a message destination
- Not accessed through interfaces
- Does not have data or state
- A single instance can process messages from multiple clients
- Message handling
  - Session and entity beans: Synchronously
  - Message-driven bean: Asynchronously

Message-Driven Beans



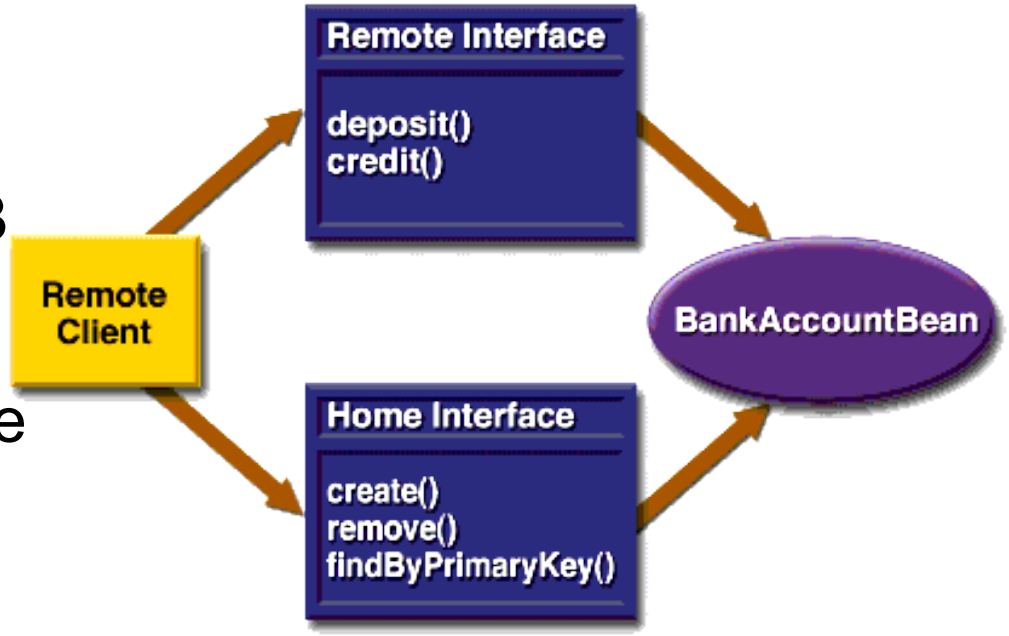
- EJB Container creates a pool of instances
- Passes context information to get them to ready state

Client Access

- Type of client accesses (remote, local, Web service)
- Remote Client
  - Can run on a different JVM
  - Can be a Web component, app client or an EJB
  - Location of accessed EJB is transient
- To allow remote acces
  - Use @Remote annotation for business interface and class

Remote Client

- Remote interface
  - Business methods of EJB
- Home interface
  - Life-cycle methods (Create and remove)
  - Finder methods (Locate entity beans)
  - Invoked on all instances of EJB



Local Client

- Runs in the same JVM
- Web component or another EJB
- Accesses the EJB knowing its location (not transparent)
- Usually an EJB with relationship to another EJB
- Local interface
  - Business methods
- Home interface
  - Finder methods; life-cycle methods

Choosing Type of Access

- Coupling of related beans
  - StudentBean and CourseBean are tightly coupled
- Type of client
  - Tightly coupled would prefer local access
  - Application clients can only do remote access
- Component distribution
  - Server-side components can exist on different machines
- Performance
  - Remote calls are slower
  - Distribution improves performance

Contents of an EJB

- Deployment descriptor: XML file that describes the EJB; persistence type, attributes
- Enterprise bean class: Compiled class
- Interfaces: Remote (or local) and home interfaces
- Helper classes: Need to be accessed; exception classes
- Stored in EJB JAR file
- One or more EJB JARs are packaged into EAR file
- Deploy EAR → Deploy EJB on the Application Server

Naming Conventions

Item	Syntax	Example
Enterprise bean name (DD)	<*name*>Bean	AccountBean
EJB JAR display name (DD)	<*name*>JAR	AccountJAR
Enterprise bean class	<*name*>Bean	AccountBean
Home interface	<*name*>Home	AccountHome
Remote interface	<*name*>	Account
Local home interface	<*name*>LocalHome	AccountLocalHome
Local interface	<*name*>Local	AccountLocal
Abstract schema (DD)	<*name*>	Account