Distributed System Using Java 2 Enterprise Edition (J2EE)

B.Ramamurthy

Introduction

- Sun Microsystems provides specifications for a comprehensive suite of technologies to solve large scale distributed system problems.
- This suite is the Java 2 Enterprise Edition, commonly known as J2EE.
- In this discussion we will discuss the architecture of J2EE and how it can be used to develop distributed multi-tiered applications.
- This discussion is based on the tutorial by Sun Microsystems Inc.

J2EE Suite

- J2EE (Java2 Enterprise Edition) offers a suite of software specification to design, develop, assemble and deploy enterprise applications.
- It provides a distributed, component-based, loosely coupled, reliable and secure, platform independent and responsive application environment.

J2EE Suite (contd.)

- Core technology: Container infrastructure, language and environment support
- XML technology
  - The Java API for XML Processing (JAXP)
  - The Java API for XML-based RPC (JAX-RPC)
  - SOAP with Attachments API for Java (SAAJ)
  - The Java API for XML Registries (JAXR)
- Web Technology
  - Java Servlets
  - JavaServer Pages
  - JavaServer Pages Standard Tag Library
- Enterprise Java Bean (EJB) technology
  - Session beans
  - Entity beans
    - Enterprise JavaBeans Query Language
  - Message-driven beans
- Platform services
  - Security
  - Transactions
  - Resources
  - Connectors
  - Java Message Service

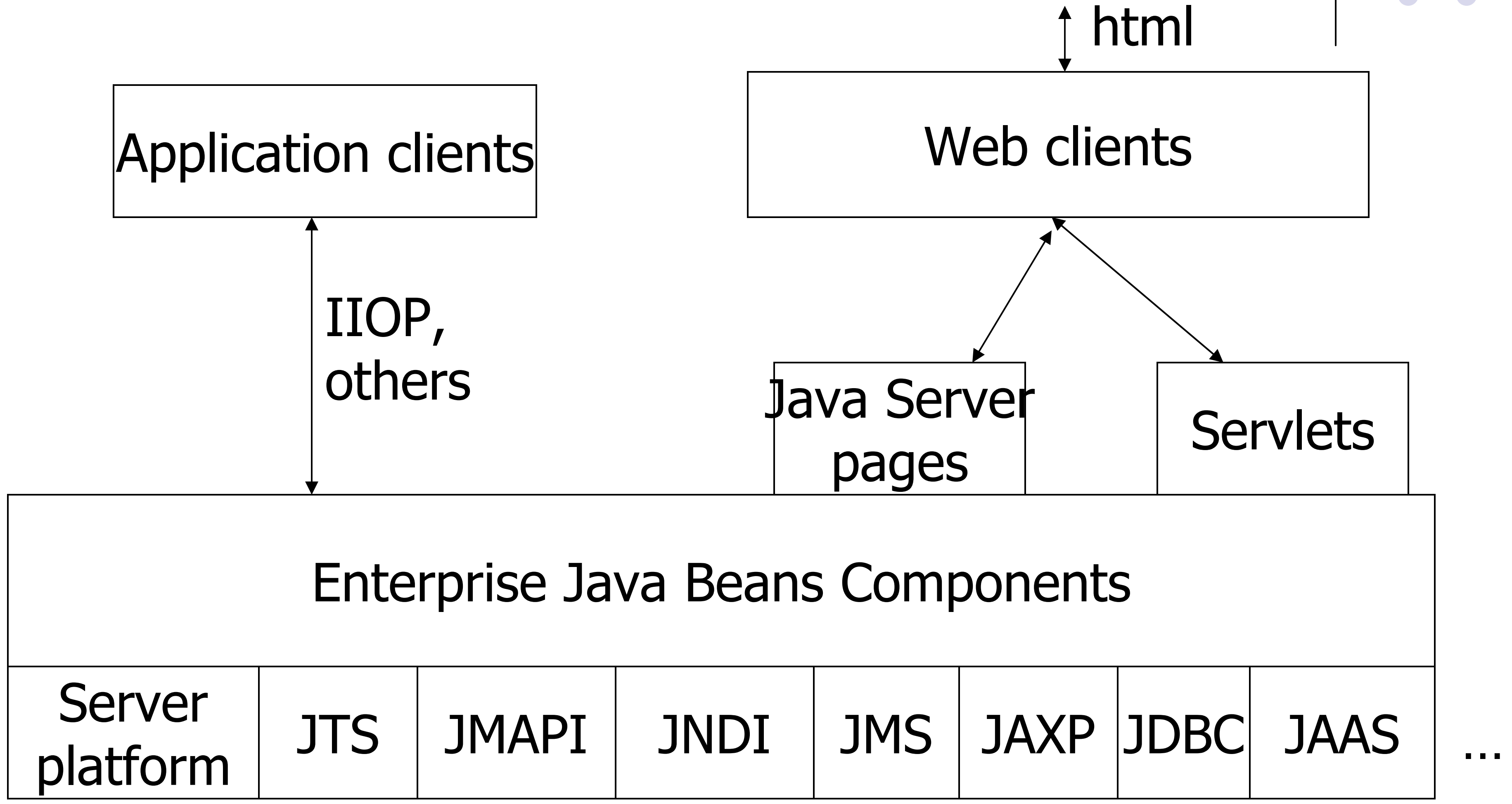
Distributed Multi-tiered Applications

- Services, clients (people and application) and data are distributed geographically across many platforms and many machines.
- Multiple tiers:
  - Client-tier (browser or client-application)
  - Web-tier (web-server: Java Server Pages)
  - Business-tier (logic; Examples: Enterprise Java Beans)
  - Enterprise-Information-System (EIS) tier (database)

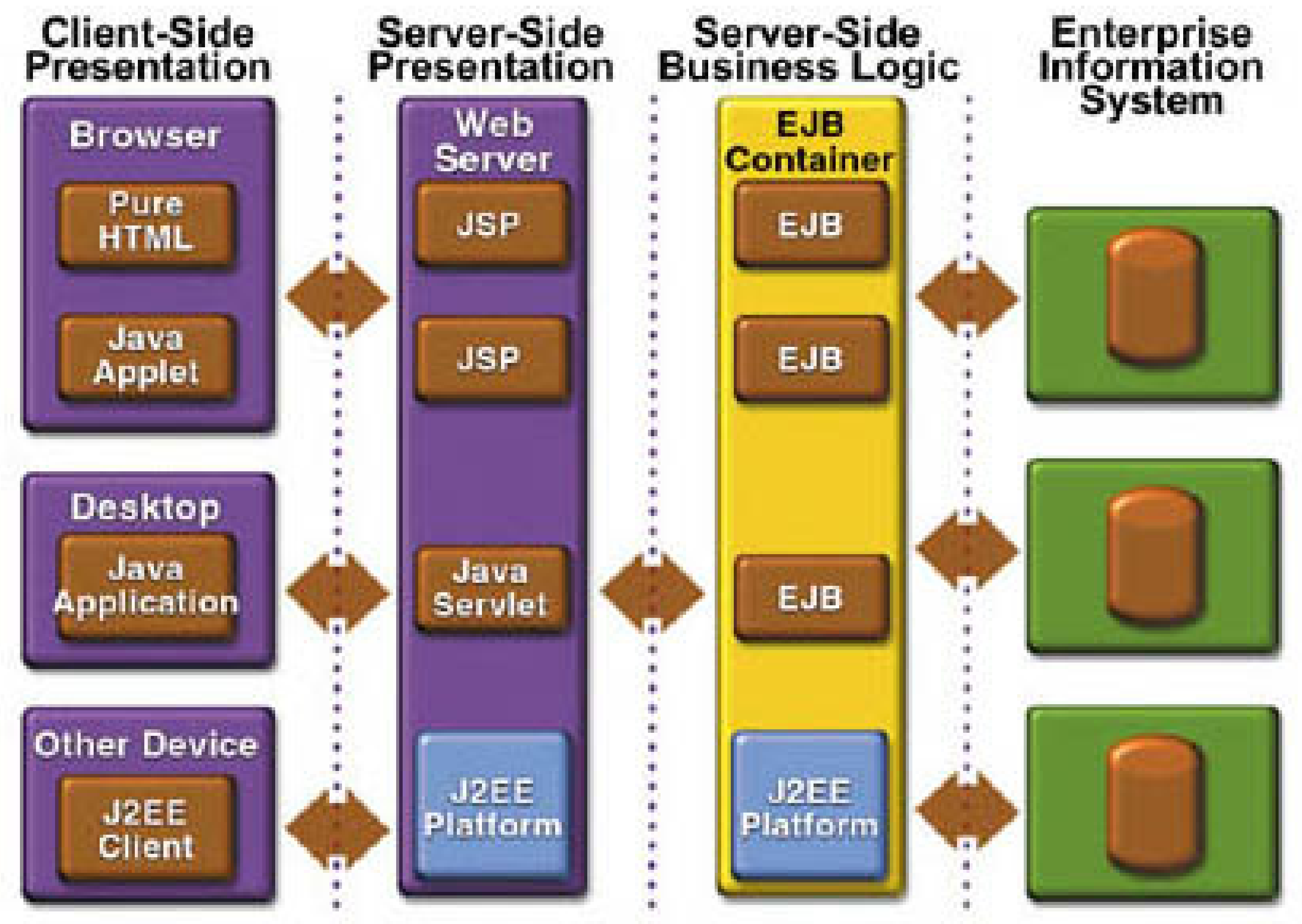
J2EE-based Application

- The J2EE APIs enable distributed systems and applications through the following:
  - Unified application model across tiers with enterprise beans
  - Simplified response and request mechanism with JSP pages and servlets
  - Reliable security model with JAAS
  - XML-based data interchange integration with JAXP
  - Simplified interoperability with the J2EE Connector Architecture
  - Easy database connectivity with the JDBC API
  - Enterprise application integration with message-driven beans and JMS, JTA, and JNDI

J2EE Technology Architecture



Enterprise Application Model



J2EE clients

- Web clients
  - Dynamic web pages with HTML, rendered by web browsers.
  - Can include applets.
  - Communicates with server typically using HTTP.
- Application clients
  - User interface using GUI components such as Swing and AWT.
  - Directly accesses the business logic tier.

Web-tier Components

- Client can communicate with the business tier either directly or through servlets ot JSP that are located in the web-tier.
- Web-tier can help in pre-processing and allows distribution of the functionality.
- Servlets are special classes to realize the request-response model (get, post of HTTP).
- JSP is a developer-friendly wrapper over the servlet classes.

Business-tier Components

- This is defined by the logic that pertains to the (business) application that is being developed.
- Enterprise Java Beans (EJB) can be used to implement this tier.
- This tier receives the data from the web-tier and processes the data and sends it to the EIS-tier and takes the data from the EIS and sends it to the web-tier.

Enterprise Information System (EIS) Tier

- In general this corresponds to the database (relational database) and other information management system.
- The other information management systems may include Enterprise Resource Planning (ERP) and legacy system connected through open database connectivity.

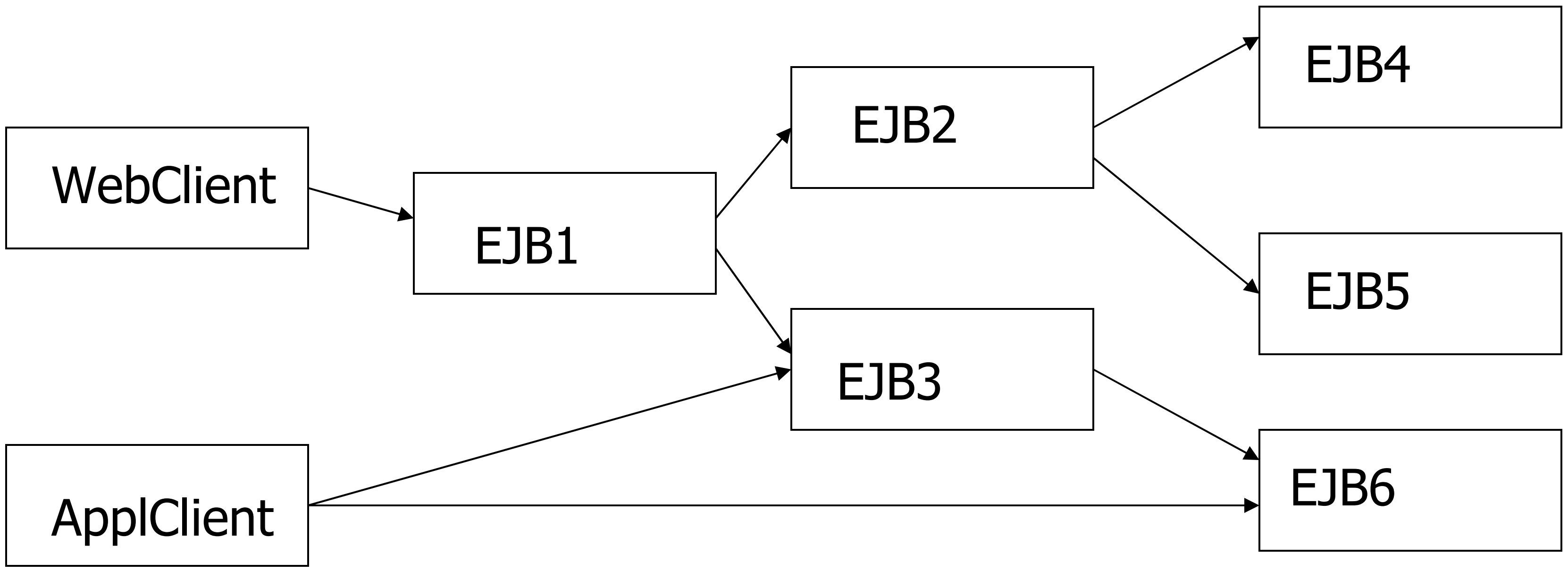
Enterprise Java Bean(EJB)

- An *enterprise bean* is a server-side component that contains the business logic of an application. At runtime, the application clients execute the business logic by invoking the enterprise bean's methods.
- Main goal of Enterprise Java Bean (EJB) architecture is to free the application developer from having to deal with the system level aspects of an application. This allows the bean developer to focus solely on the logic of the application.

Enterprise Java Bean (EJB) (contd.)

- Deployable unit of code.
- At run-time, an enterprise bean resides in an EJB container.
- An EJB container provides the deployment environment and runtime environment for enterprise beans including services such as security, transaction, deployment, concurrency etc.
- Process of installing an EJB in a container is called EJB deployment.

Enterprise Application with many EJBs



Lets consider a shopping front application and figure out the possible components (EJBs)

Roles in EJB Development

- Bean developer: Develops bean component.
- Application assembler: composes EJBs to form applications
- Deployer: deploys EJB applications within an operation environment.
- System administrator: Configures and administers the EJB computing and networking infrastructure.
- EJB Container Provider and EJB server provider: Vendors specializing in low level services such as transactions and application mgt.

Types of Enterprise Java Beans

- Session beans
- Entity Beans
  - Bean-managed Persistence (BMP)
  - Container-managed Persistence (CMP)
  - Enterprise Javabeans Query Language
- Messaging Bean
  - Session bean with Java Messaging features

Session Beans

- For transient functions
- Represents “conversational” state
- Typically one per request
- Data is non-persistent
- Lifetime is limited by the client’s: once the client exits, the session bean and data are gone.
- Simple and easy to program.
- Light-weight.

Entity Bean

- “Transactional” in behavior
- Can be shared among clients
- Persistent: data exists permanently after client quits.
- Corresponds to a row a relational database.
- The persistence (storing into the database) can be automatically done by the “container” (CMP) or explicitly by the bean (BMP)

Entity Bean (contd.)

- Data is at the heart of most business applications.
- In J2EE applications, entity beans represent the business objects that need persistence (need to be stored in a database.)
- You have choice of bean-managed persistence (BMP) and container-managed persistence (CMP).
- In BMP you write the code for database access calls. This may be additional responsibility but it gives control to the bean developer.

Message-Driven Bean

- A message driven bean is an enterprise bean that allows J2EE applications to process messages asynchronously.
- It acts as a JMS listener, which is similar to an event listener except that it receives messages instead of events.
- The messages can be sent by any J2EE component: an application client, another enterprise bean, or a web component, or a non-J2EE system using JMS.
- Retain no data or conversational state.

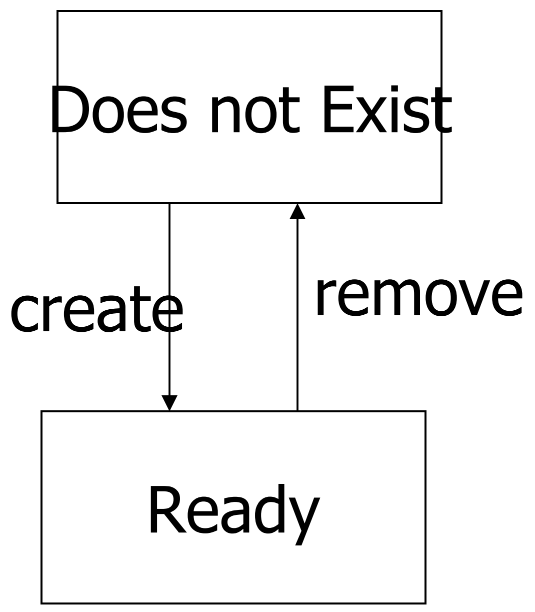
The life cycles of enterprise beans

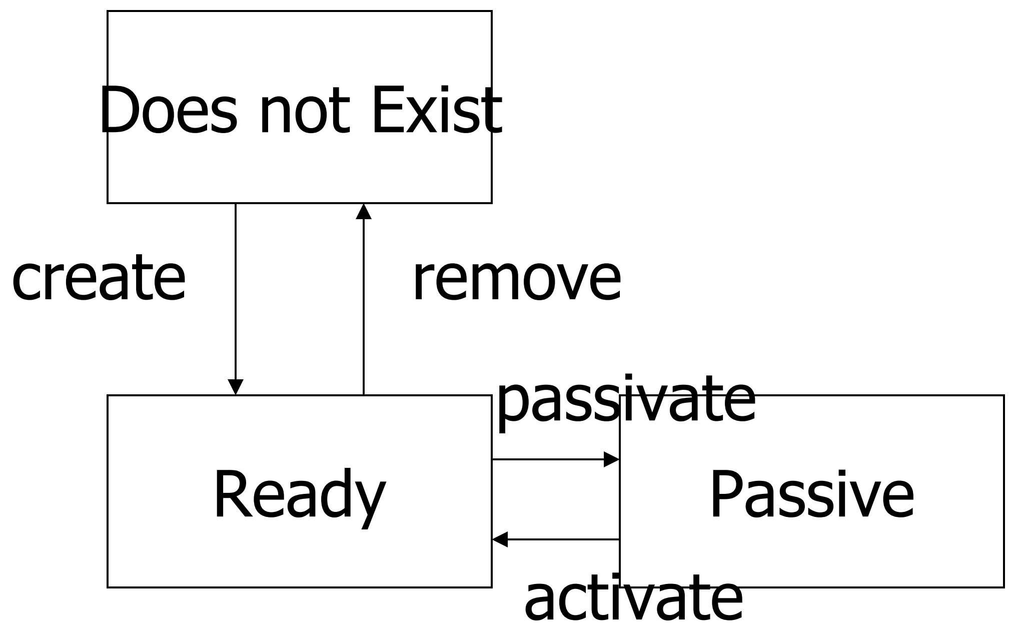
- An enterprise bean goes through various stages during its lifetime. Each type has different life cycle.

Life Cycle Differences

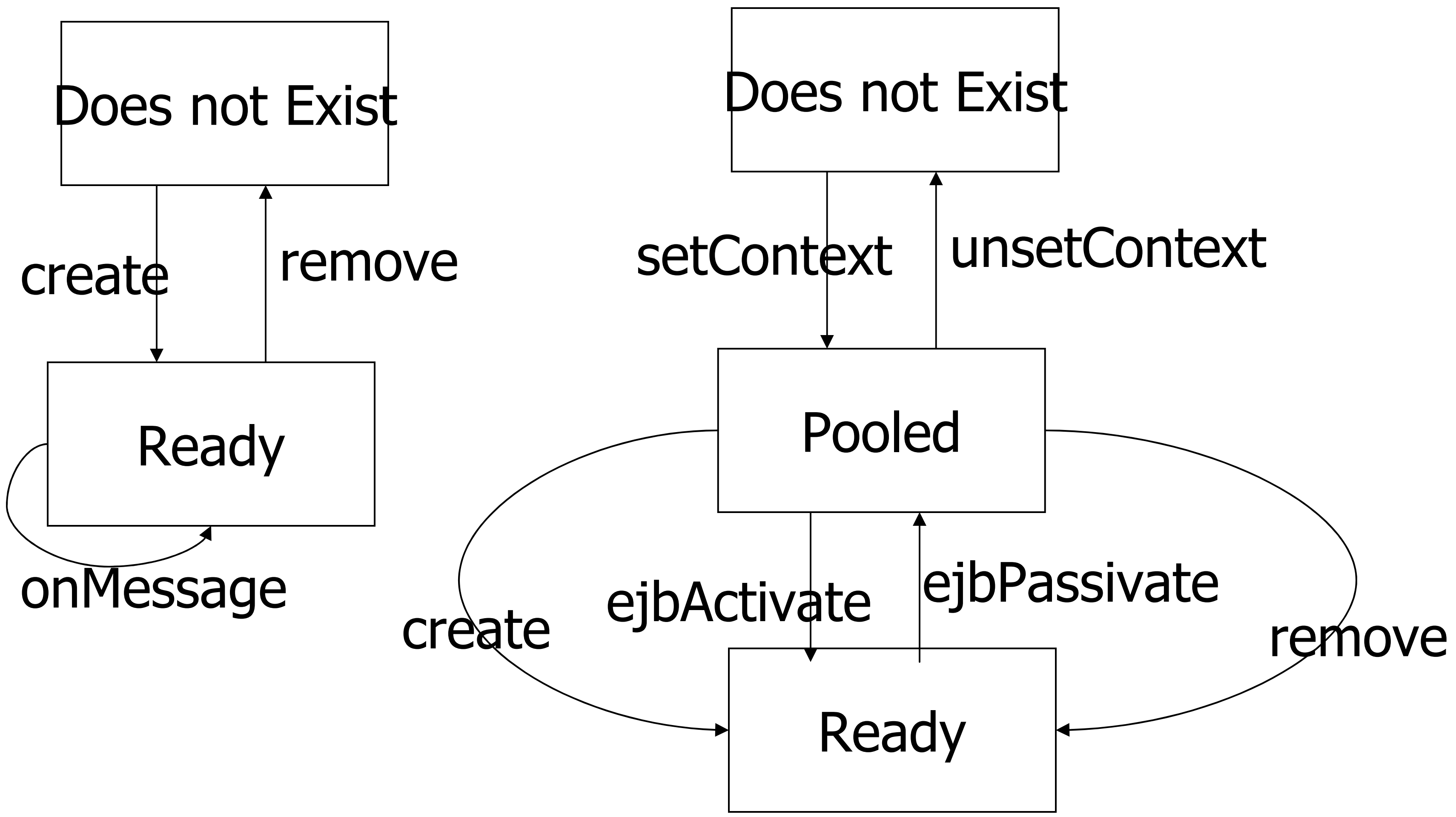
	Session Bean	**Entity Bean**
Object state:	Maintained by container	Maintained by DB
Object Sharing:	No sharing: per client	Shared by multiple client
State Externalization:	State is inaccessible to other programs	Accessible to other programs
Transactions:	Not recoverable	State changed transactionally and is recoverable.
Failure Recovery:	Not guaranteed to survive failures	Survives failures and restored when the container restarts.

Session bean





Message-driven Bean and Entity-Bean Lifecycle



Business Entities, Processes and Rules

- EJB Applications organize business rules into components.
- Components typically represent a business entity or business process.
- Entity: is an object representing some information maintained in the enterprise. Has a “state” which may be persistent.
- Example: Customer, Order, Employee,
- Relationships are defined among the entities: dependencies.

Process

- Is an object that typically encapsulates an interaction of a user with business entities.
- A process typically updated and changes the state of the entities.
- A business process may have its own state which may exist only for the duration of the process; at the completion of the process the state ceases to exist.
- Process state may be transient or persistent.
- States ate transient for conversational processes and persistent for collaborative processes.

Rules

- Rules that apply to the state of an entity should be implemented in the component that represents the entity.
- Rules that apply to the processes should be implemented in the component that represents the processes.

Choosing Entity or Session Bean

- Entity (business entity) is typically implemented as entity bean or a dependent object of an entity bean.
- Conversational (business) process as a session bean.
- Collaborative bean as an entity bean.
- Any process that requires persistence is implemented as an entity bean.
- When exposure to other applications are not needed for an entity or process (local/private process) then they are implemented as bean dependent objects.

Contents of an Enterprise Bean

- Interfaces: The remote and home interface for remote access. Local and local home accesses for local access.
- Enterprise bean class: Implements the methods defined in the above interfaces.
- Deployment descriptor: An XML file that specifies information about the bean such as its type, transaction attributes, etc.
- Helper classes: non-bean classes needed by the enterprise bean class such as utility and exception classes.

Enterprise Bean Parts

<<Home Interface>>
AccountHome

create()
find()
remove()

<<Remote Interface>>
Account

debit()
credit()
getBalance()

<<Enterrpise Bean class>
AccountBean

ejbCreate()
ejbFind()
ejbRemove()
debit()
credit()
getBalance()

Deployment Descriptor

name = AccountEJB
class = AccountBean
home = AccountHome
remote = Account
type = Entity
transaction = required
…..

Container Services

- A container interfaces the programmatic components such as EJBs to the declarative components.
- Container services include security, transaction management, naming services, and remote connectivity.
- The fact that the J2EE architecture provides configurable services means that application components can behave differently based on where they are deployed.
- The concept of “deployable units” and “containers” where they can be deployed is central to J2EE.

Compilation and Deployment

- Compilation (building the executables) uses build tool such as Apache Ant.
- The components of the various tiers are packaged: .jar, .war, .ear
- Declarative resources are added.
- A deploy tool or management tool is used to deploy the packaged units into a J2EE server (container).

Summary

- J2EE environment provides a framework for bundling together the components into an application and provide the applications necessary common services such as persistence, security, mail, naming and directory service etc.
- Next class we will look a complete running example.
- Browse through:
    - http://java.sun.com/j2ee/faq.html
    - http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/index.html#chapters
    - http://java.sun.com/developer/onlineTraining/J2EE/Intro2/j2ee.html