TUTORIAL ON EJB

Enterprise JavaBeans™

Phillip Bride

GemStone Systems, Inc.

GEMSTONE

Topics

- Introducing EJB
- Overview of EJB
- Summary

Roots of EJB

- Sun Microsystems, Inc.
  - Enterprise JavaBeans Specification 1.0
- Many contributing partner companies
  - IBM, Oracle, GemStone, BEA
  - EJB roundtables occur quarterly
- Build the server-side of JavaBeans

What is EJB?

> A component architecture for development and deployment of object-oriented distributed enterprise-level Java applications.
>
> -- Sun Microsystems, EJB Spec

A server-based **Framework** to build modular Java applications

Other descriptions

- Anne Thomas, Patricia Seybold Group

> EJB extends the JavaBeans component model to support server components.
>
> EJB takes the concept of “Write once, run anywhere to a new level.”
>
> Most organizations have not felt scalability pressures that required a multi-tier architecture…until Web-based computing.

- From articles in NC World by Rawn Shaw

> EJB is industrial-strength Java.
>
> A Bean can serve just about any function.
>
> Application developers can take a set of beans and wire them together into an application.
>
> Keep in mind, EJB is a framework.

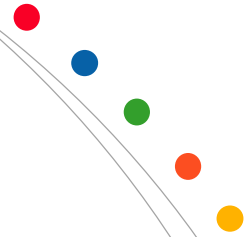
Companies involved in EJB

- Sun Microsystems + Partners
- Standards committees
- EJB Vendors

The Goals of EJB

- Standard component architecture
- Easy-to-write applications
- Address development, deployment and runtime issues
- Interoperate

Without EJB Framework



Business issues

- Monolithic programs are expensive
- Internet demands flexibility
- Dynamic business structures demand flexibility

Technical issues

- Custom code lengthens design cycle
- Difficult to reuse code
- Much of the infrastructure gets re-invented

No need to re-invent infrastructure

Without EJB Framework

Developers build

- Threading & process control
- Distributed execution control
- Transactional control
- State management
- Resource management
- Security
- **Interfaces**
- **BUSINESS LOGIC**

Using EJB Framework

Developers build

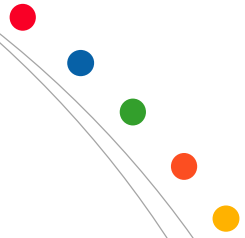
- **Interfaces**
- **BUSINESS LOGIC**

EJB Framework supplies

- Threading & process control
- Distributed execution control
- Transactional control
- State management
- Resource management
- Security

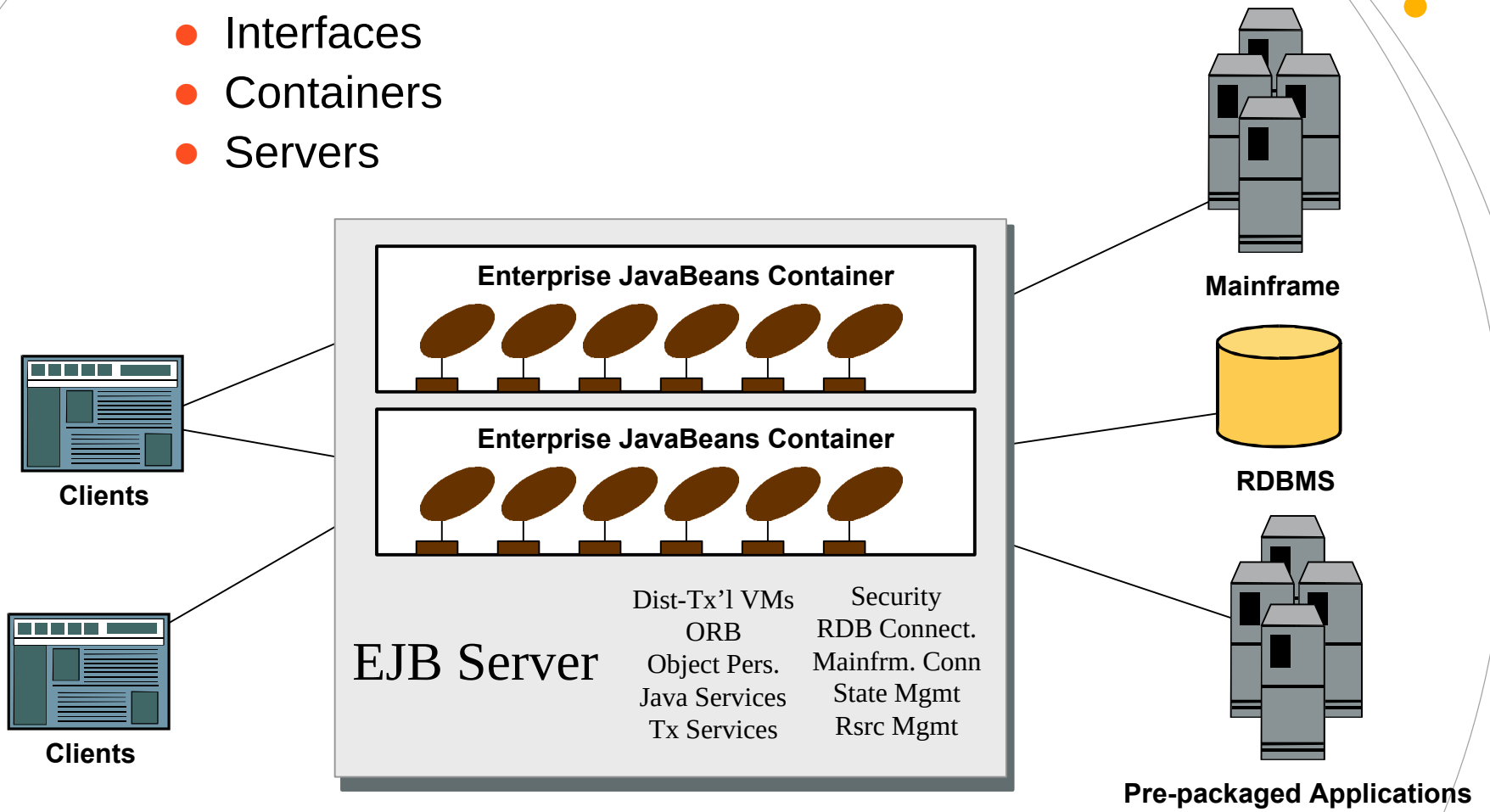
Topics

- Introducing EJB
- **Overview of EJB**
- Summary



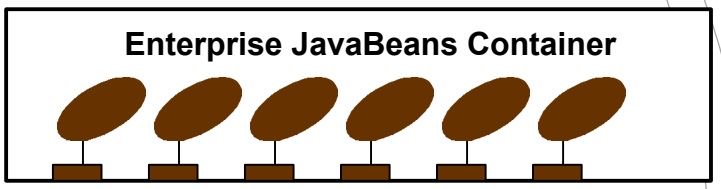
Components in EJB Framework

- Four main components are:
  - EJBs
  - Interfaces
  - Containers
  - Servers



What are EJBs

- Application components
  - Java Classes
    - Persistent entities
    - Client services, e.g. Business logic
  - Run inside containers
    - Data access
    - Logical extension of client program (stateless)
    - Client conversation spanning multiple client requests (stateful)



EJB = Enterprise bean = Bean = Enterprise JavaBean

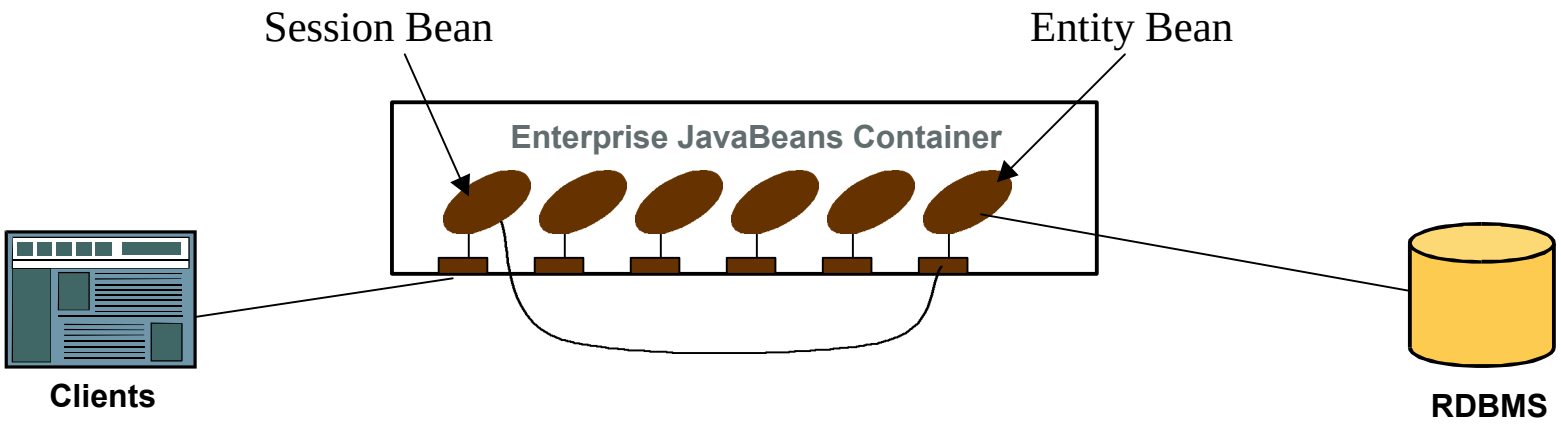
Types of Beans

Session Beans

- Interact with clients
- Model business Logic
- Short-lived

Entity Beans

- Represent persistent data
- Long-lived



Session Beans

Stateful

Assigned to 1 client only

Keeps info about a client

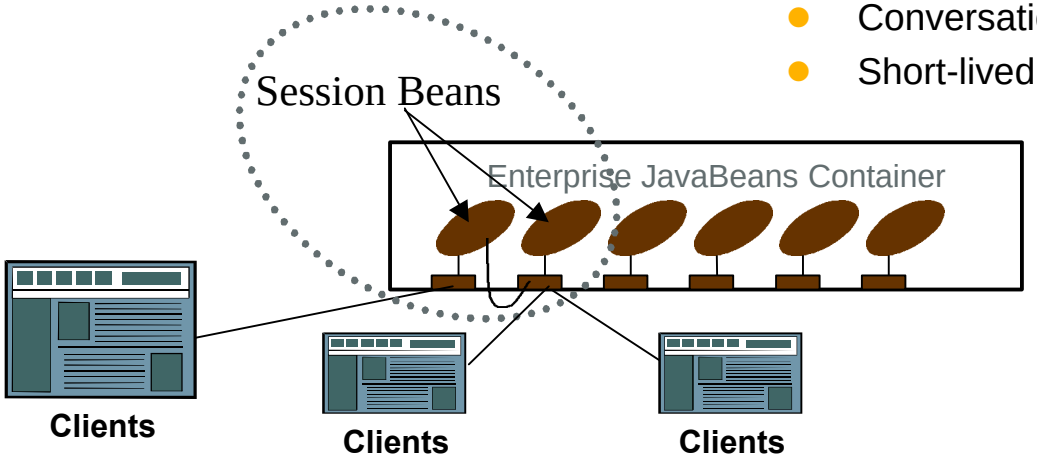
- Has attributes for client's state
- Assigned to a client for lifetime
- Can touch and cache persistent data
- Serialized methods, no loopbacks
- Non-persistent
- Short-lived (client, timeout, server crash)

Stateless

Many clients can access it

Implements business logic

- Can service multiple clients
- Lifetime controlled by container
- No state across methods or Tx
- Each instance identical upon creation
- Touches persistent data
- State managed by bean
- Conversational state managed by client
- Short-lived



Stateful Session Bean

ASSIGNED TO ONE CLIENT

Session Bean definition

Interface to context

Info about Client (state)

Hold bean context

```
import javax.ejb*;
import java.rmi.*;
import java.util.*;

public class CartBean implements SessionBean {
    private SessionContext ctx;

    private Properties env;
    private Customer customer;
    private ItemList items;

public void setSessionContext (SessionContext ctx)
   throws RemoteException {
    this.ctx = ctx
    }
```

Stateful Session Bean

Container callbacks
- Create
- Remove
- Passivate
- Activate

Business Method

```
public void ejbCreate (String customerName)
   throwsRemoteException,NotFoundException {
     create bean instance
     initializes state about customer
     }
public void ejbRemove() throws RemoteException {
     release resources if necessary
     }
public void ejbPassivate()throws RemoteException {
     close databases if open
     release sockets if open
     store persistent data somewhere if needed
     }
public void ejbActivate()throws RemoteException, {
     acquire resources if needed
     }

public ItemList getItems(String type)throws
   RemoteException, {
     QUERY DATA STORE
     MODIFY CLIENT STATE
     return items;
     }
}
```

Stateless Session Bean

BUSINESS LOGIC

Session Bean definition

Constants

Non-conversational state

Hold bean context for bean lifetime

```
import javax.ejb*;
import java.rmi.*;
import java.util.*;

public class BusLogicBean implements SessionBean {

    private final static DataStore dataStore =
       “AcmeStore”;

    private SessionContext ctx;
    private Properties env;


public void setSessionContext (SessionContext ctx)
   throws RemoteException {
    }
```

Stateless Session Bean

Container Callbacks
- Create
- Remove
- Passivate
- Activate

Business Methods

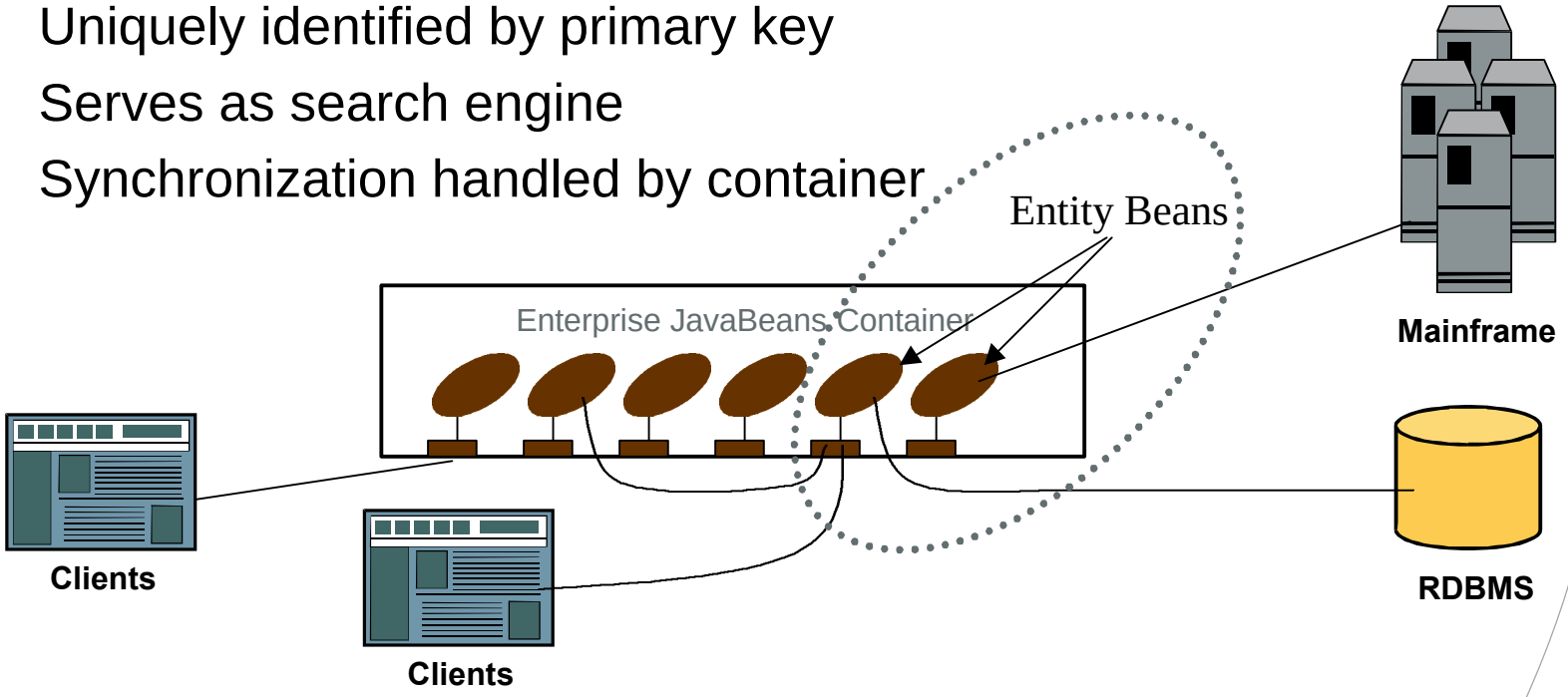
```
public void ejbCreate()throws RemoteException {
    }
public void ejbRemove() throws RemoteException {
    }
public void ejbPassivate()throws RemoteException {
    }
public void ejbActivate()throws RemoteException, {
    }

public ItemList purchaseItems(String customerName,
   ItemList items)throws RemoteException {
    BUSINESS LOGIC
    return a value;
    }
public ItemList businessLogic(String vendorName,
   ItemList items)throws RemoteException, {
    BUSINESS LOGIC
    return a value;
    }
}
```

Entity Beans

Object representation of persistent data

- Used to create, cache, update & remove data in data stores
- Persistent
- Shared by multiple clients
- Uniquely identified by primary key
- Serves as search engine
- Synchronization handled by container



Entity Beans & Persistence

Bean Managed

Developer writes persistence code

Developer

- Determines data to persist
- Initializes data
- Describes finder methods
- Writes data store accesses
- Writes data mapping

Container

- Synchronizes state
- Manages pool of beans
- Manages lifecycle

Container Managed

Developer focuses on data use

Developer

- Specifies data to persist
- Initializes attributes
- Describes finder methods

Container

- Generates data store accesses
- Maps data
- Synchronizes state
- Manages pool of beans
- Manages lifecycle

Entity Bean

PERSISTENT DATA

Entity Bean definition

Data items

Hold bean context for bean lifetime

```
import javax.ejb*;
import java.rmi.*;
import java.util.*;

public class ItemBean implements EntityBean {

    public String name;
    public float price;
    private EntityContext ctx;
    private DataStore dataStore;

public void setEntityContext (EntityContext ctx) throws
   RemoteException {
    this.ctx = ctx;
    this.dataStore
      =this.ctx.getEnvironment().getProperty("dataStore");
    }
public void unsetEntityContext() throws
   RemoteException{}
```

Entity Bean

Container Callbacks
- Create
- Remove
- Passivate
- Activate

- Load
- Store
- Find

```
public void ejbCreate(String name, float
   price)throws RemoteException {
   this.name = name;
   this.price = price;
    }
public void ejbRemove() throws RemoteException {}

public void ejbPassivate()throws RemoteException {}
public void ejbActivate()throws RemoteException {}




public void ejbLoad()throws RemoteException {}
public void ejbStore()throws RemoteException {}
public void ejbFindByPrimaryKey(ItemKey key) throws
   RemoteException, FinderException {
   get data out of data store by primary key lookup
   }
```

Entity Bean

Business Methods

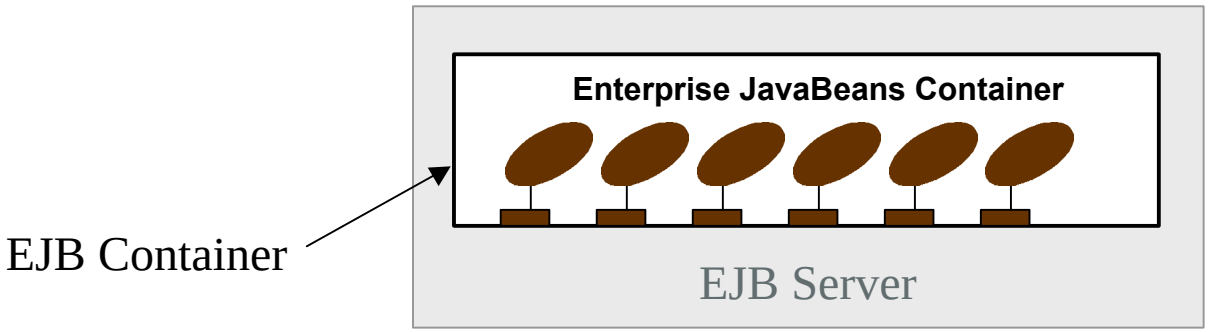
```
public float getPrice(float saleValue) {
     open data store;
     get primary key;
     get data;
     close data store;
     return this.data;
     }

}
```

The Containers

A context within which to run Beans

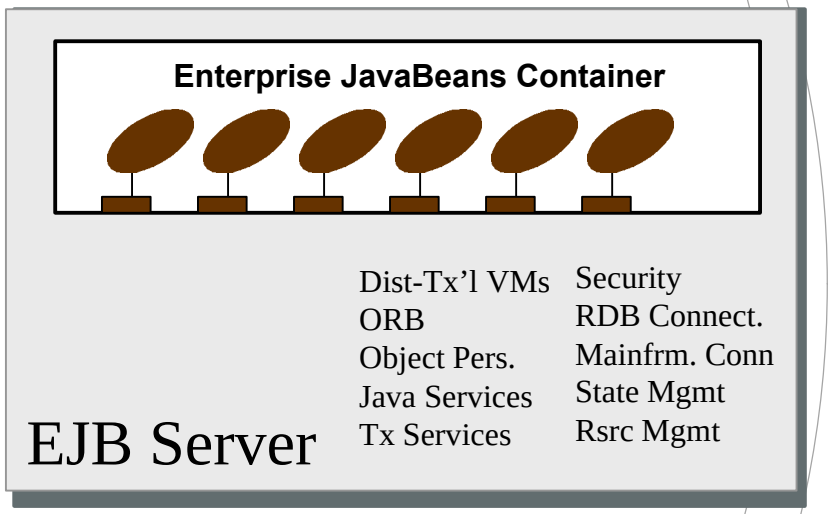
- Containers exist within EJB servers
- Manage
  - Pools of Beans
  - Bean lifecycles
  - Interfaces between clients and beans
  - Manages state (CMP)
  - Threads for beans
  - Communication to EJB server for lower-level services



The EJB Server

Low-level infrastructure to manage containers

- Services provided by server
  - Distributed, transactional VMs
  - ORB
  - Java object persistence
  - Java services
  - Security
  - RDB connectivity
  - Mainframe connectivity
  - State management
  - Resource management

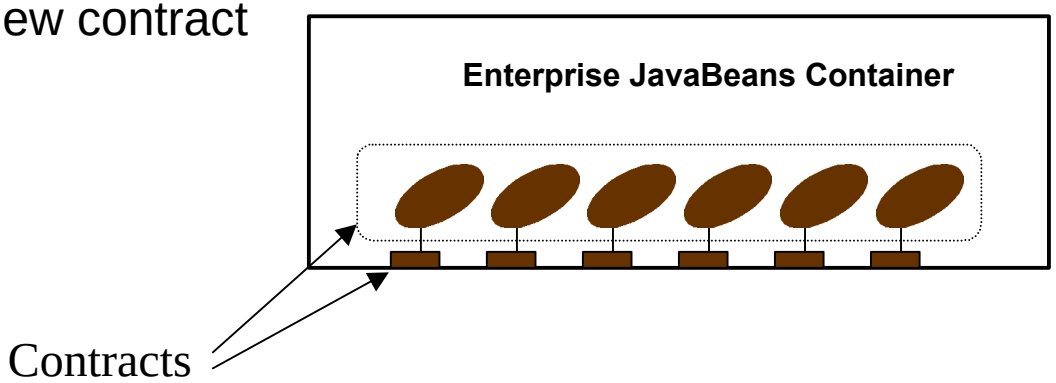


The Contracts

- EJB developers have two basic questions:

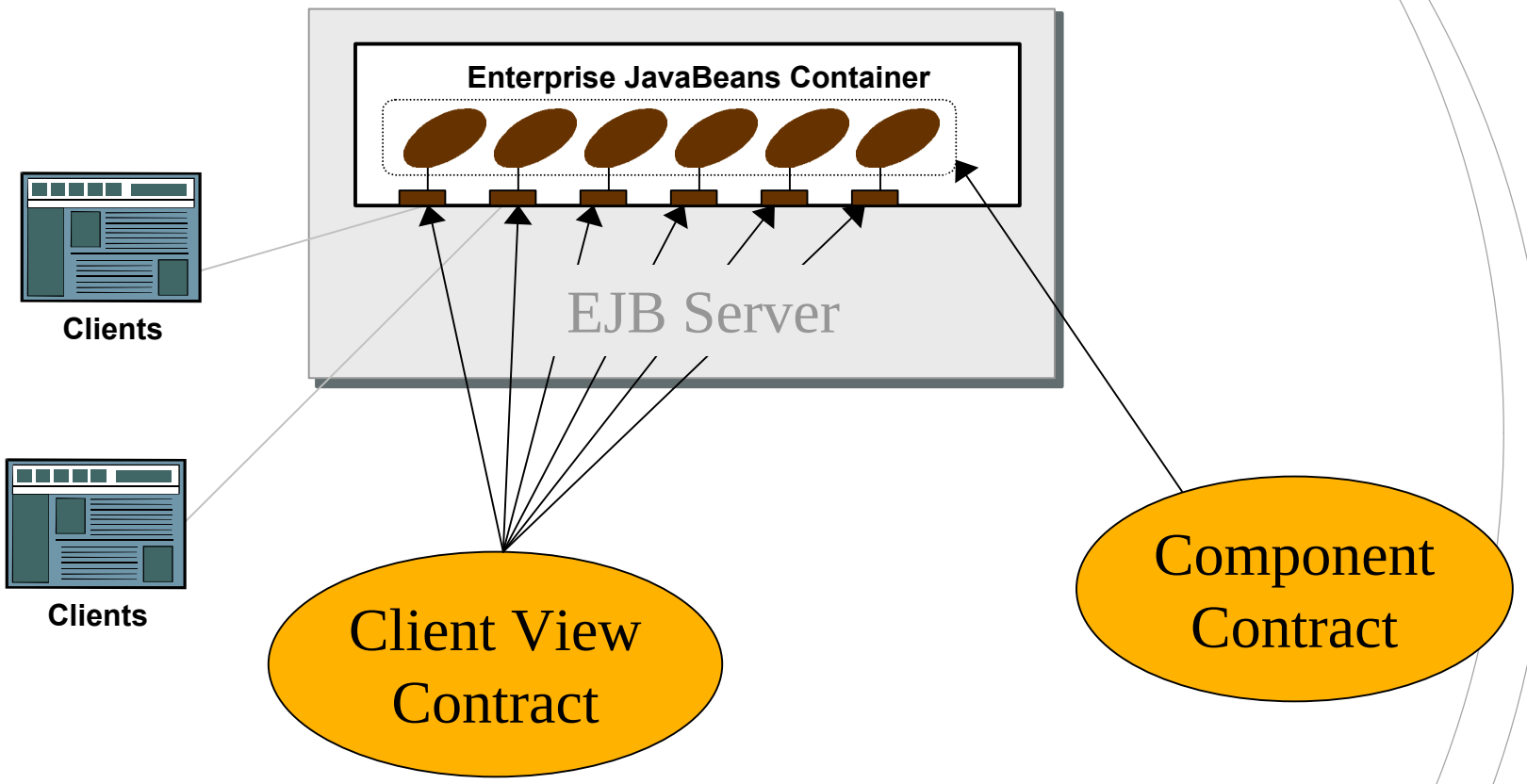
If I write an EJB,

- How can I be assured that any CONTAINER can use it?
  - Component contract
- How can I be assured that any CLIENT can use it?
  - Client view contract



Contracts implemented by Interfaces

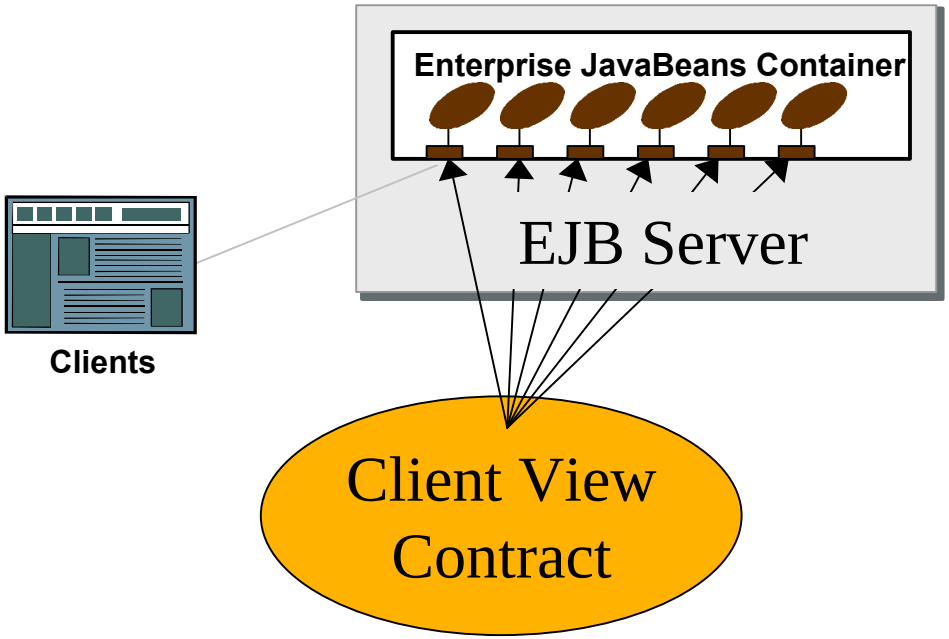
Standard interfaces for beans used to interact with clients and containers



Client View Contract

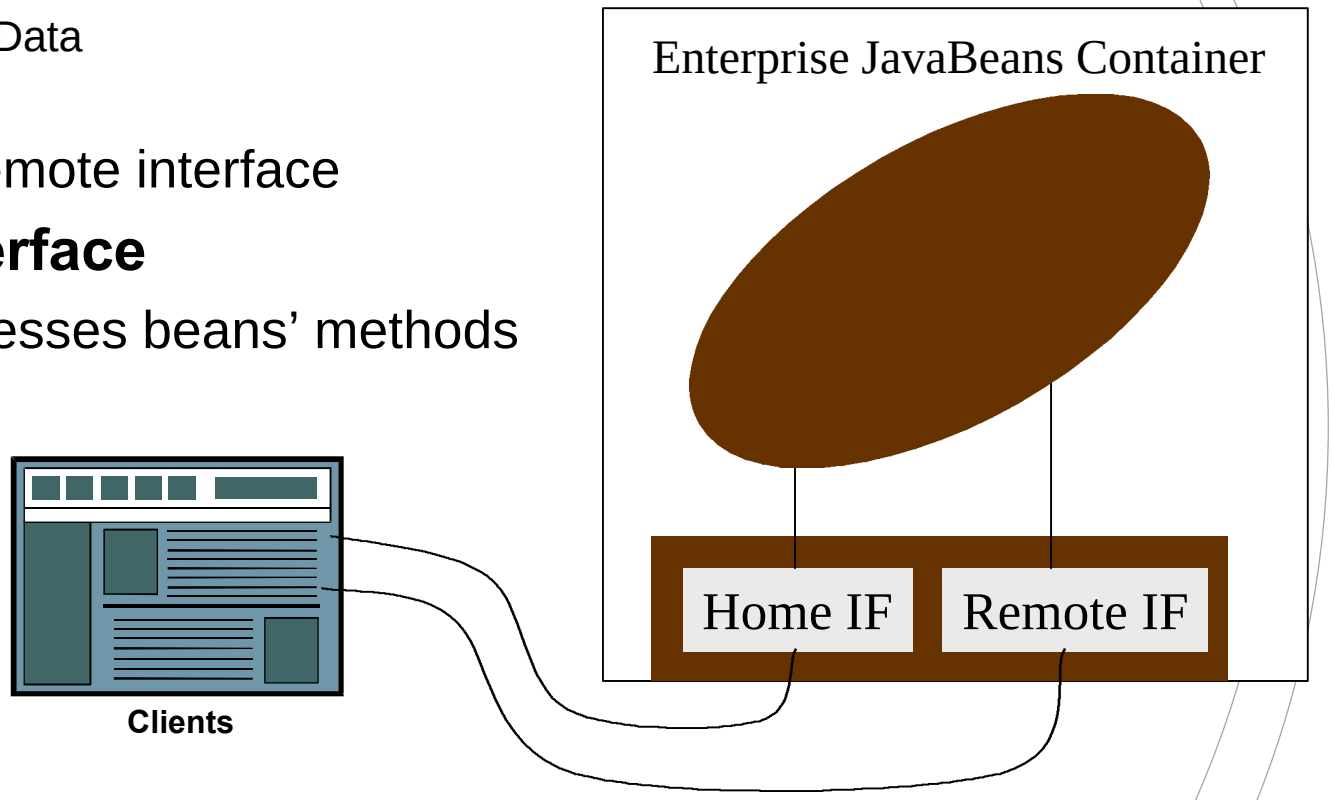
Standard for client-bean communication

- Consists of bean's home & remote interfaces
- Client interacts with bean through home & remote interfaces
- Interfaces delegate messages to beans



Client View Contract

- **Home Interface**
  - Client first accesses bean
  - Client manages bean
    - create/remove
    - getMetaData
    - find
  - Creates remote interface
- **Remote Interface**
  - Client accesses beans' methods



Home Interface

- Provides methods for Client
  - create, remove, business rules, find (entity)
- Create and find methods return Remote Interfaces for the bean

Stateless Session Bean Home Interface

```
Import java.rmi.*;
import javax.ejb.*;

public interface BusLogicHome extends EJBHome {
    public EJBMetaData getEJBMetaData() throws RemoteException;
    public void remove (Handle handle) throws RemoteException,
                                              RemoveException;
    pulbic void remove (Object primaryKey) throws RemoteException,
                                              RemoveException;
    public BusLogic  create()  throws         RemoteException,
                                              CreateException;
}
```

Remote Interface

- Client invokes business methods
- Unique object identity
  - Implicit ID for session beans
  - Unique primary key for entity beans

Stateless Session Bean Remote Interface

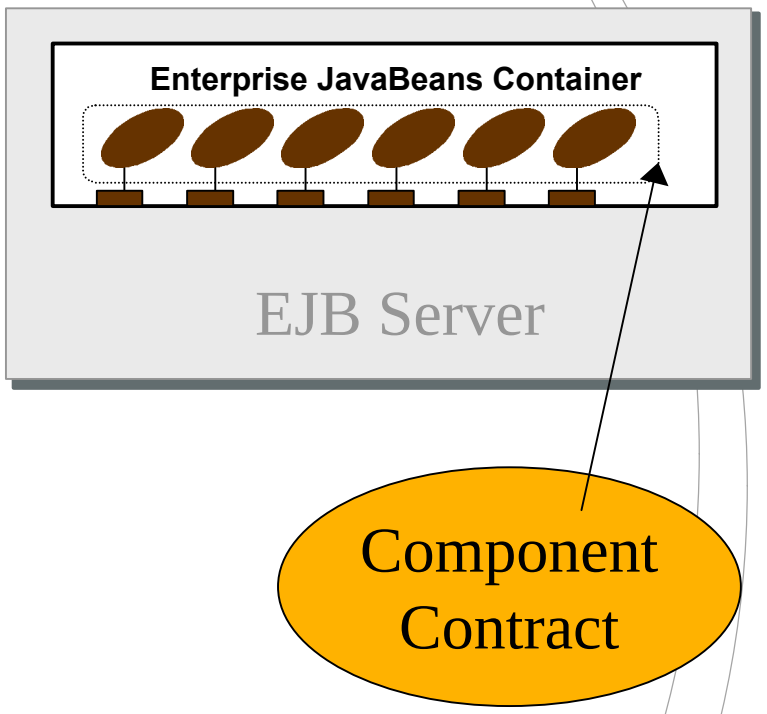
```
Import java.rmi.*;
import javax.ejb.*;

public interface BusLogic extends EJBObject {
     public EJBHome        getEJBHome()              throws RemoteException;
     public Handle         getHandle()               throws RemoteException;
     public Object         getPrimaryKey()  throws RemoteException,
     public Boolean        isIdentical(EJBObject otherObj)
                                                     throws   RemoveException;
     pulbic void remove(Object primaryKey)  throws RemoteException,
                                                          RemoveException;
     public String businessLogic(String vendorName, ItemList items)
                                                     throws   RemoteException;
     public String purchaseItems(String customerName, ItemList items)
                                                     throws RemoteException;
}
```

Component Contract

Standard for bean-container communication

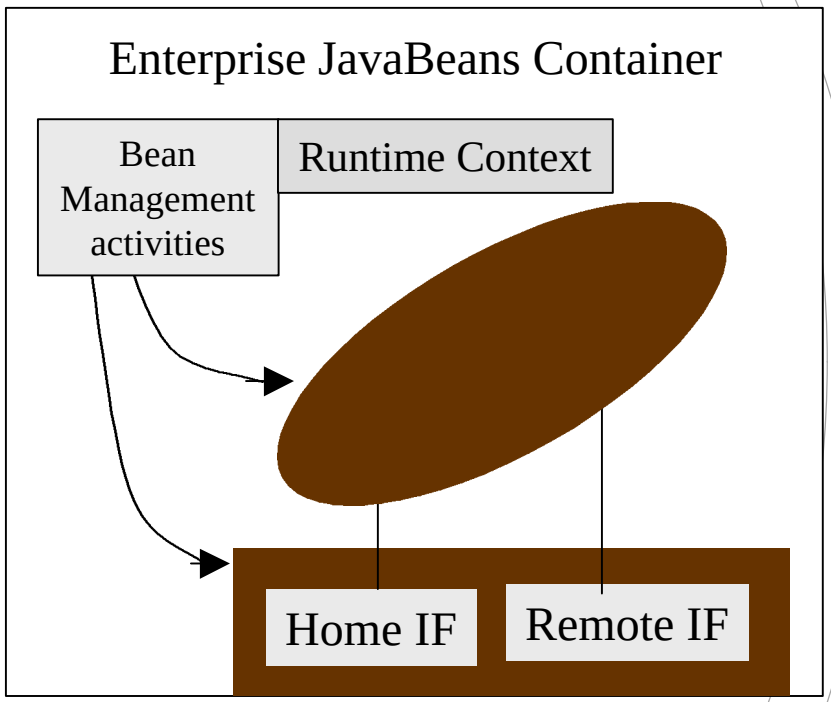
- Lets container manage EJB
  - Life cycle (callbacks)
  - Access to business objects
  - Object Management
- Provides services to EJB
  - Location
  - Transactions
  - Security



Component Contract

Container

- Implements bean's home & remote interfaces
- Wraps client requests with container services
- Provides security, transaction, environment
- Delegates messages to bean.
- Manages beans
- Provides runtime context
- Notifies bean of lifecycle events



Component contract

Calls from the container various Beans

```
CartBean.ejbRemove();

ItemBean.ejbPassivate();

BusLogicBean.ejbActivate();



ItemBean.ejbLoad();
ItemBean.ejbStore();
ItemBean.ejbFindByPrimaryKey(ItemKey key);
```

Deployment Descriptor

- Defines attributes of the bean
  - Bean home name (JNDI name)
  - Environment properties
  - Home interface class name
  - Remote interface class name
  - Bean class name
  - Bean type (Stateless, stateful, entity)
  - Container-managed fields
- Used by the container to create I/Fs
- Can be modified

ejb-jar file

- A standard format to package Beans and deployment information.
  - JAR file manifest
  - Java class files
  - Deployment descriptors
  - Environment properties

Bean lifecycles

- Stateful Session Bean example
  - Client
    - looks up bean
    - Sends home interface a message
  - Home Interface
    - issues a create message
  - Container
    - Sends the bean a newInstance() message
    - Sends the bean a setSessionContext(ctx) message
    - Passes back a Remote Interface, EJBObject
    - Sends the bean an ejbCreate(args) message (initializes)

EJB development activities

Deploy

Application Assembled

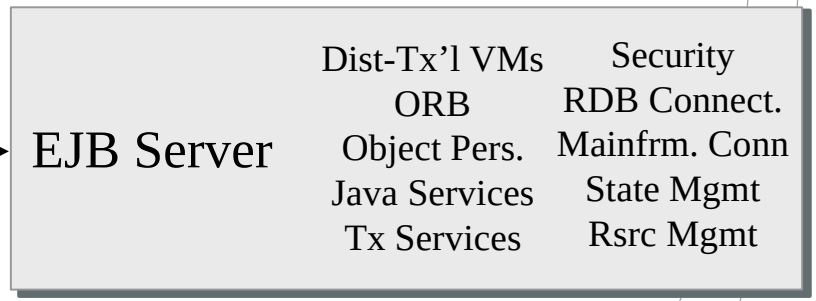
Enterprise Bean

EJB Container

EJB Server

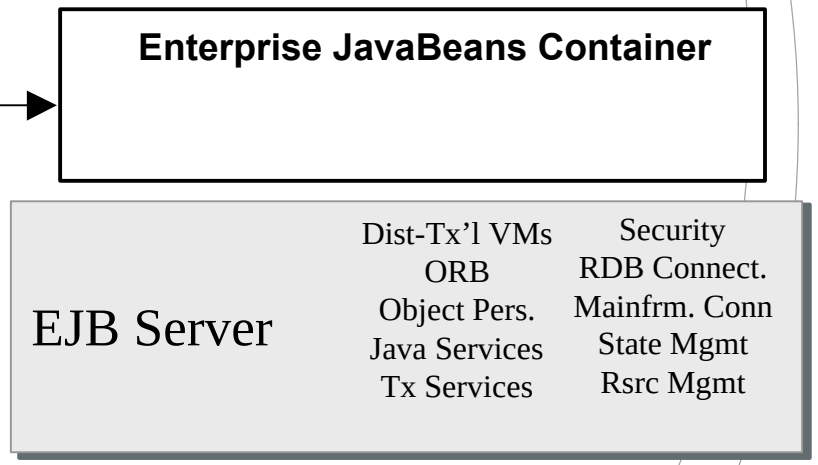
EJB development roles

- **EJB Server Provider**



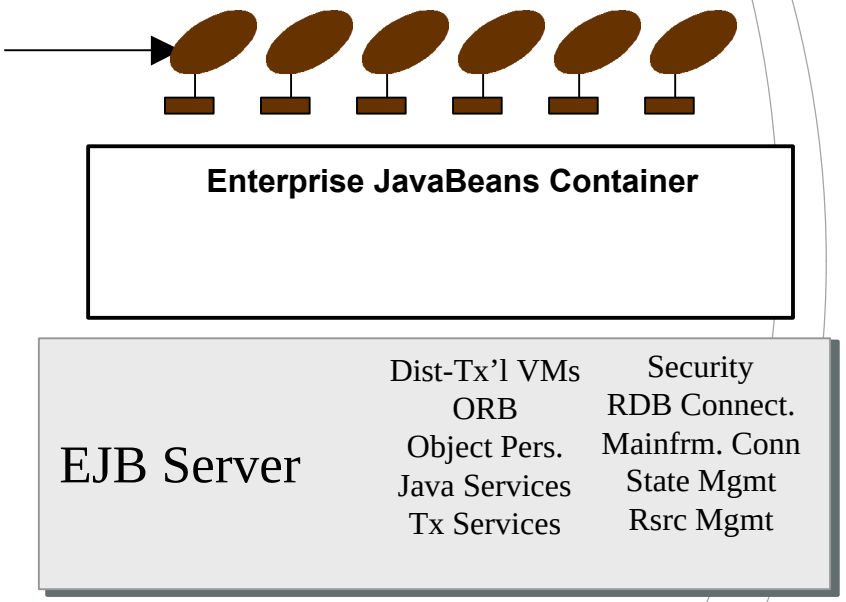
EJB development roles

- **EJB Container Provider**
- EJB Server Provider



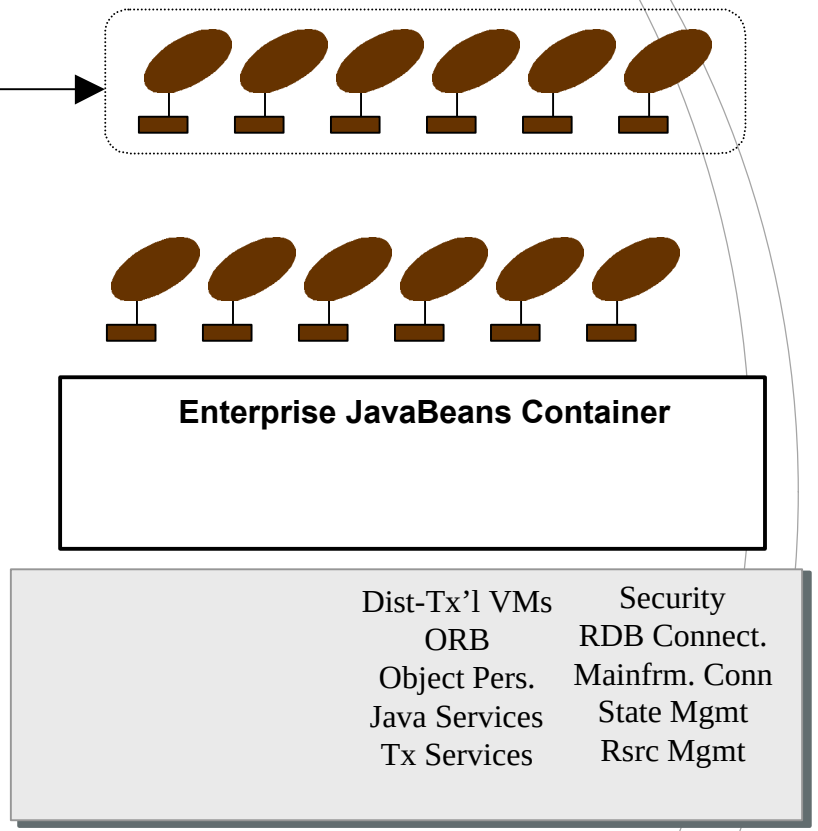
EJB development roles

- **Enterprise Bean Provider**
- EJB Container Provider
- EJB Server Provider



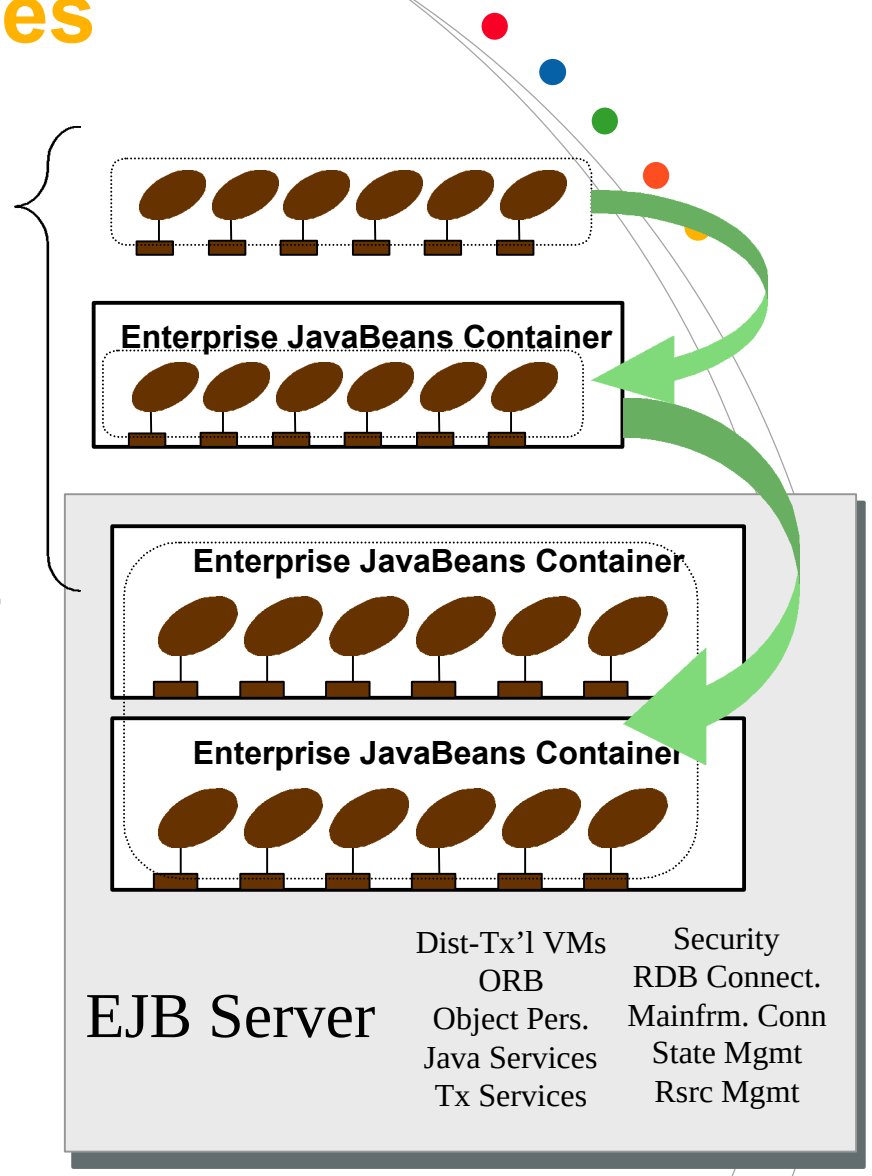
EJB development roles

- **Application Assembler**
- Enterprise Bean Provider
- EJB Container Provider
- EJB Server Provider



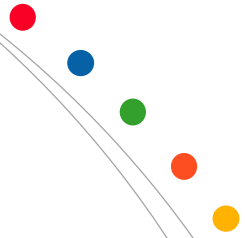
EJB development roles

- **Deployer**
- Application Assembler
- Enterprise Bean Provider
- EJB Container Provider
- EJB Server Provider



Topics

- Introducing EJB
- Overview of EJB
- **Summary**



EJB Roadmap

- Version1.0 3/98
  - First Release
- Draft 2.0 release 12/98
  - Clarifications & fix errors
  - Updates
- Draft 2.0 release 1H99
  - Add new features
- Version 2.0 mid 99
  - Entity beans required
  - JMS support
  - General availability

Summary

- A Framework that provides many services
- Lets you focus on business logic
- No need to re-invent infrastructure
- Standards based