

# **Java based middlewares**

## **JINI, EJB, J2EE**



**ICS 237- Distributed Systems  
Middleware**

**Nalini Venkatasubramanian**

# Java



- “*lingua franca*” to easily interconnect data and application across organizational boundaries.
- Java-based middlewares
  - Application servers -- BEA WebLogic
  - Messaging products -- Active Software's ActiveWorks
  - DBMS based - exploit DBMS systems with server-based Java object execution features.
- Features
  - JVM -- Java's runtime execution environment
  - RMI, RMI registry (Java's RPC and directory service)
  - Type safety and sandboxed execution (JVM process can service multiple requests without interfering)

[Old Article on Java middleware](#)

# Java RMI

- Java communication

- Sockets

- pt-to-pt, duplex
    - Pre-determined format, protocol

- RPC

- Abstract to procedure call
    - Standard data representations

- RMI

- Object oriented RPC
    - RPC+Java serialization to pack objects

- 3 processes

- Client
  - Server
  - RMI registry

- Remote class must

- *extends Remote interface*
  - Server extends *java.rmi.server.UnicastRemoteObject*

- *rmic compiler*

- *stub and skeleton generation*

- Other Issues

- Serialization
  - Exception handling
  - Constructors

```
public interface PrintService extends Remote {  
    int print(Vector printJob) throws  
    RemoteException;  
}
```

# Java Serialization



- Store and retrieve objects
  - Capture enough state for reconstruction
  - Generate a bytestream
- Java interfaces for serialization
  - `java.io.Serializable`: default serialization mechanism
  - `java.io.Externalizable`: custom serialization
  - Serialize to file, serialize an entire class

# Jini Motivation



- Need a distributed system based on the idea of federating groups of users and the resources required by those users.
- Need an open software architecture that enables the creation of network-centric solutions which are highly adaptive to change.
- Middleware solution to build adaptive networks that are scalable, evolvable and flexible as typically required in dynamic computing environments.

# Jini – Java Middleware

- Network extension of Java
  - Users share services and resources over a network
  - Easy access to resources anywhere on the network while allowing network location of the user to change
- Simplifying the task of building, maintaining, and altering a network of devices, software, and users
  - Support true plug and play in LAN-based networked systems
  - SOHO (small office, home office environments)
  - ROHO (remote office, home office environments)
- System of federated users and resources
  - Appears to users as a single system
  - A client/resource/service may belong to more than one Jini system at a time

# Environmental Assumptions



- Existence of a network of “reasonable speed”
  - “reasonable” network latency
- Connected devices have “some memory and processing power”
  - Those that don’t must have a Jini *proxy* that does have memory and processing power
- Needs the Java Environment
  - Members are assumed to agree on basic notions of trust, administration, identification, and policy.

# Jini Advantages



## FEATURE

- Java Virtual Machine
- Portable object code
- Downloadable code
- Unified type systems

## BENEFITS

Homogeneous network  
Architecture independence  
Dynamic environment  
No impedance mismatch



# Jini Structure



<b>Jini Services</b>	<ul style="list-style-type: none"><li>• JavaSpaces™</li><li>• Transaction Managers</li><li>• Printing, Storage, Databases...</li></ul>
<b>Jini Infrastructure</b>	<ul style="list-style-type: none"><li>• Discovery</li><li>• Lookup Service</li></ul>
<b>Jini Programming Model</b>	<ul style="list-style-type: none"><li>• Leasing</li><li>• Distributed Events</li><li>• Transactions</li></ul>
<b>Java 2 Platform</b>	<ul style="list-style-type: none"><li>• Java RMI</li><li>• Java VM</li></ul>

# Jini Services



- Anything that can be used in a Jini system
  - Entity used by a person, program, another service, storage...
- Utilized through a Service Protocol
  - Set of interfaces written in Java
  - Services carry the code needed to use them
  - A small set of protocols is predefined
    - e.g. Discovery, Join, Lookup
- Communication happens through RMI
  - Allows for objects *and* code to be sent around the network
- Security: incorporates Java's security models

# What Jini is



- Services carry the code needed to use them
  - proxies are dynamically downloaded by clients when they need to use a service
- A “meta-service” provides access to all other services
  - Lookup services keeps track of all other services in a community.
    - Entries are downloadable Java objects that act as local proxies to the real service
- Bootstrapping process to find proxies for the lookup service
- Service discovery and join: service protocols that allows services (both hardware and software) to discover, become part of, and advertise supplied services to the other members of the federation

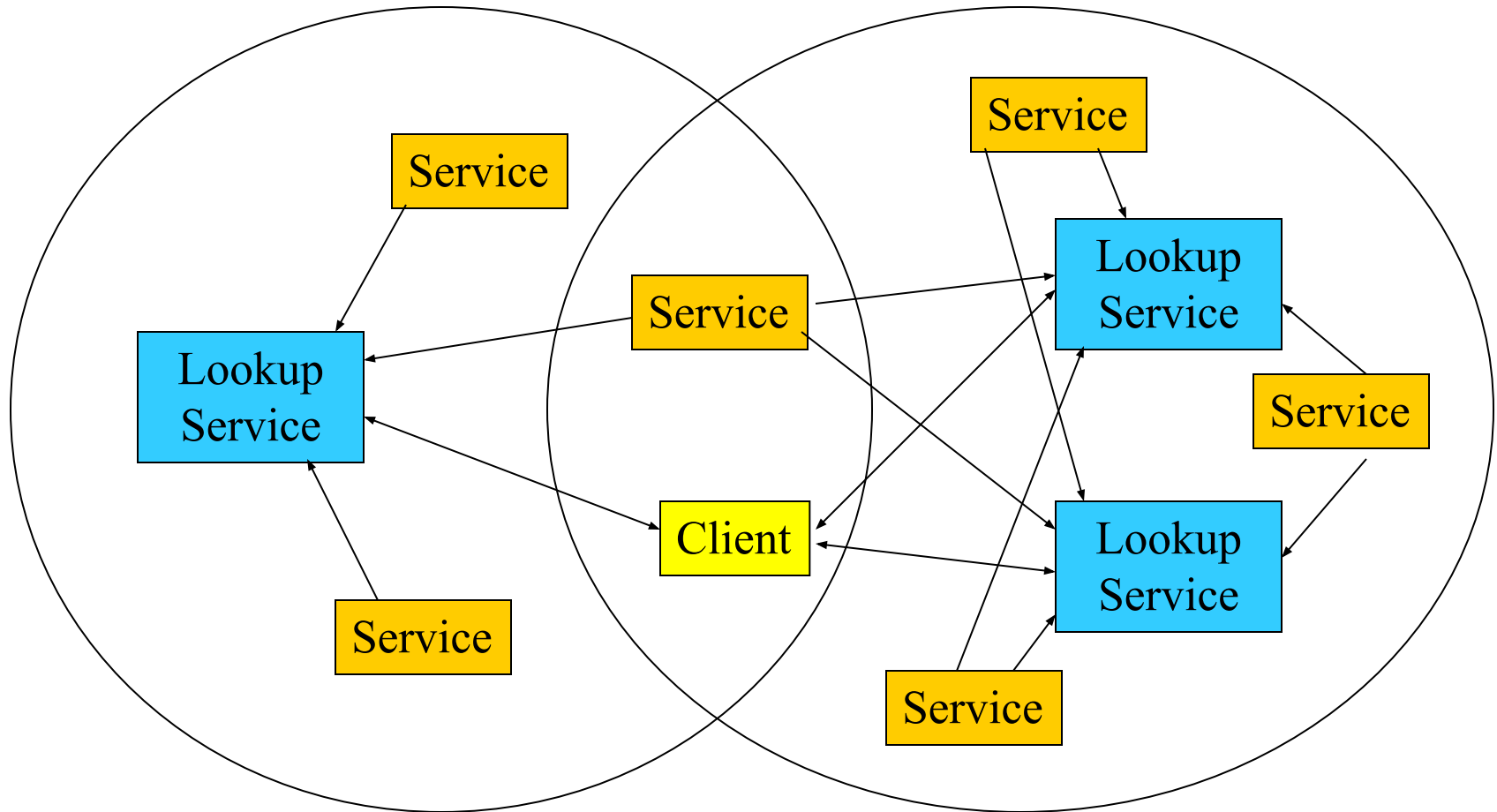
# What Jini is not



- Not just RMI
- Jini is not just a name server
- Jini is not a system consisting of client and servers. It is system consisting of services that can be collected together for the performance of a particular task.
- Jini is not JavaBeans
  - JavaBeans provides a way for software components to find and introspect each other
  - intended for use within a single address space
  - less dynamic (design-time, not runtime)
- Jini is not EJB
  - similar to Jini but intended to hook together legacy systems covered by Java wrappers to form the back-end business logic of enterprise applications



# Jini Overview



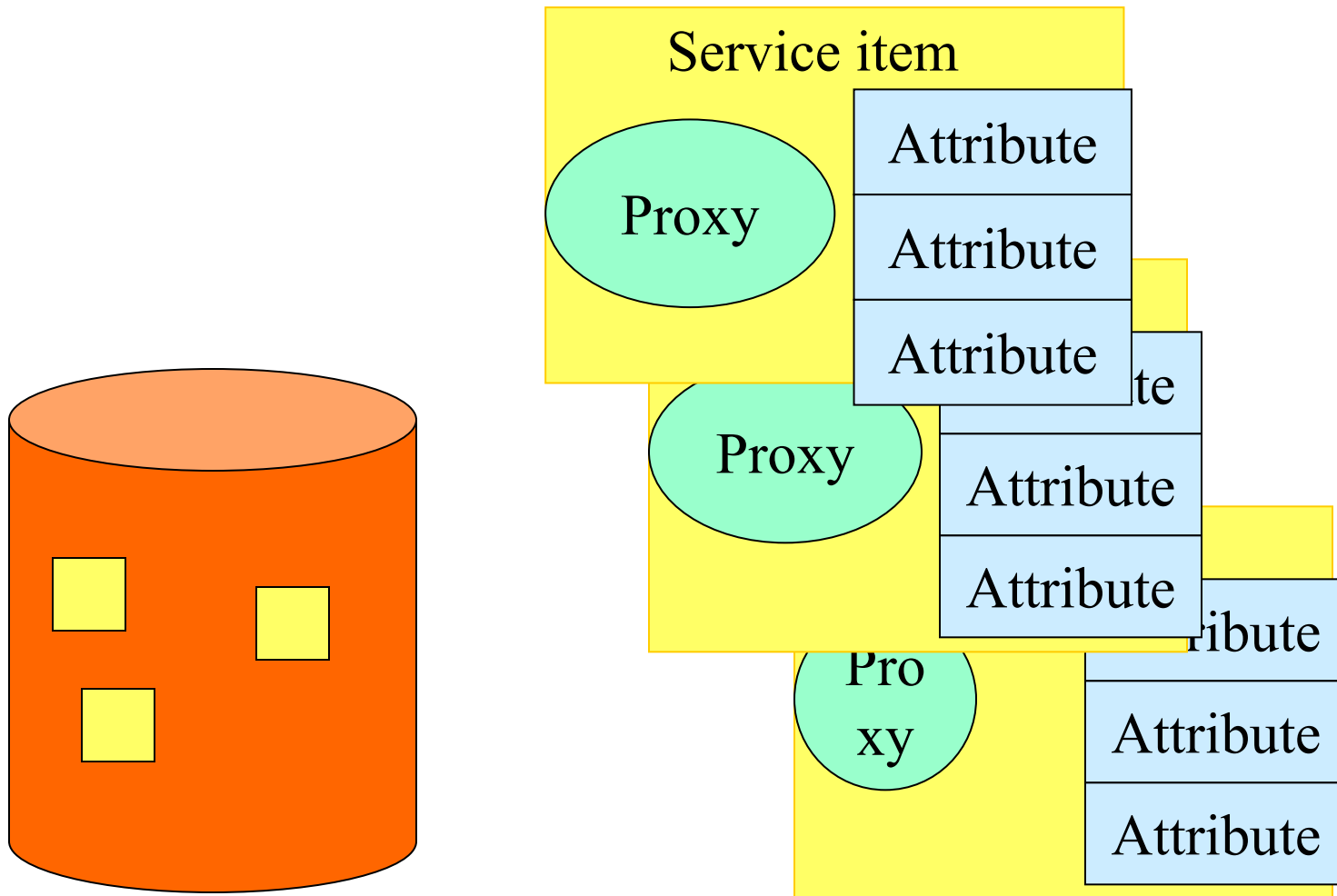
Jini group: cs237.uci.edu

Jini group: public

# Jini Lookup Service

- Core/ Central bootstrapping mechanism for the system
  - provides the major point of contact between the system and users of the system.
- Maps interfaces to objects that implement those interfaces
  - Interface only describes functionality of a service
  - Descriptive entries can be associated with a service to allow more flexible searching
- Services can appear/disappear in a lightweight way
  - A service is added to a lookup service by a pair of protocols called Discovery and Join.
  - The service locates an appropriate lookup service using the discovery protocol.
  - The service is added using the join protocol.

# Lookup Service



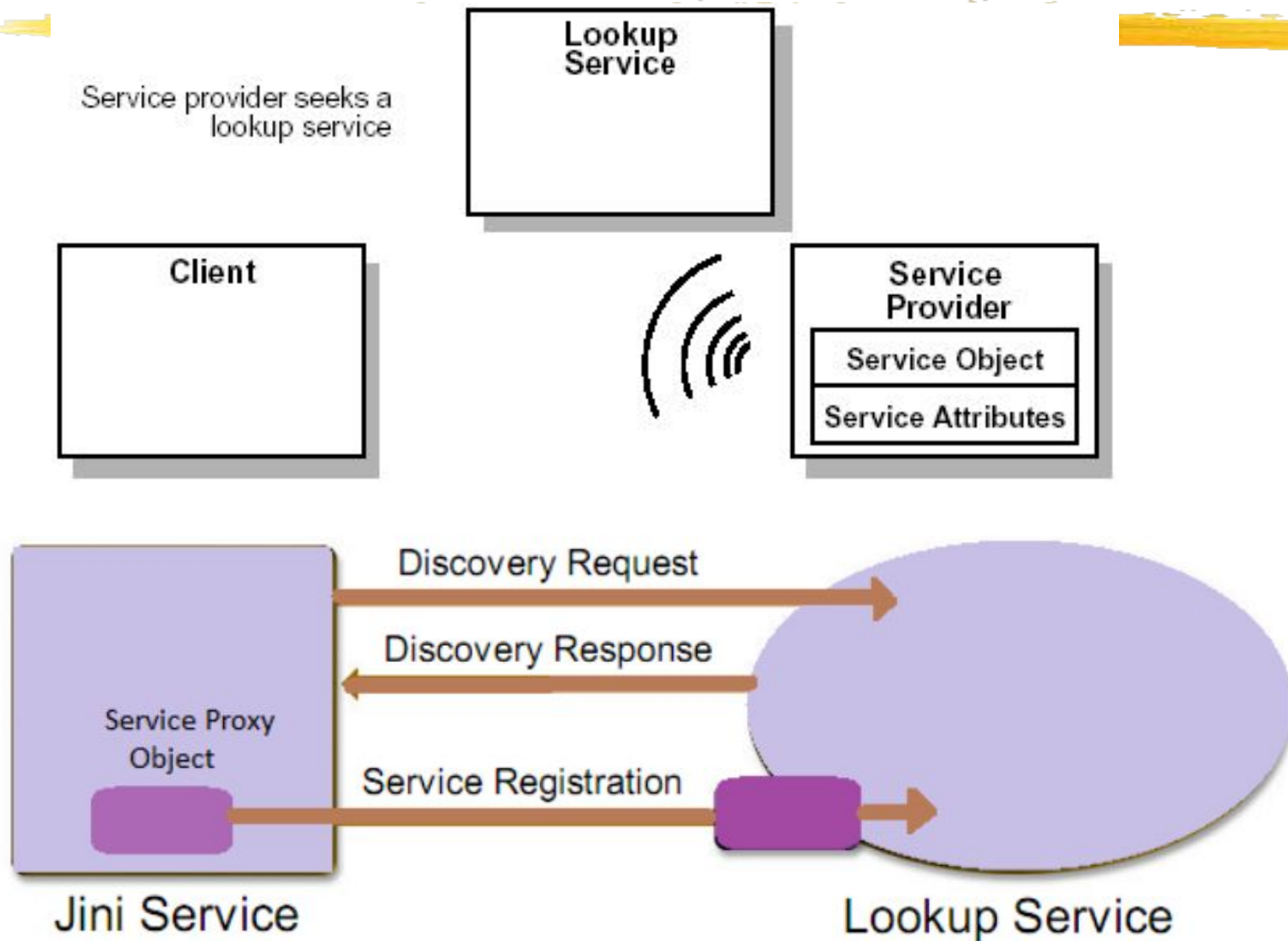
# Lookup Service



- Jini Lookup Service is an interface
  - Implementations can incorporate other lookup services
    - Hierarchical Lookup
    - Bridge between lookup services
- Discover, Join and Lookup Protocols
  - Discover to find a lookup service
  - Join to add to the lookup service
  - Lookup to find a service and use it



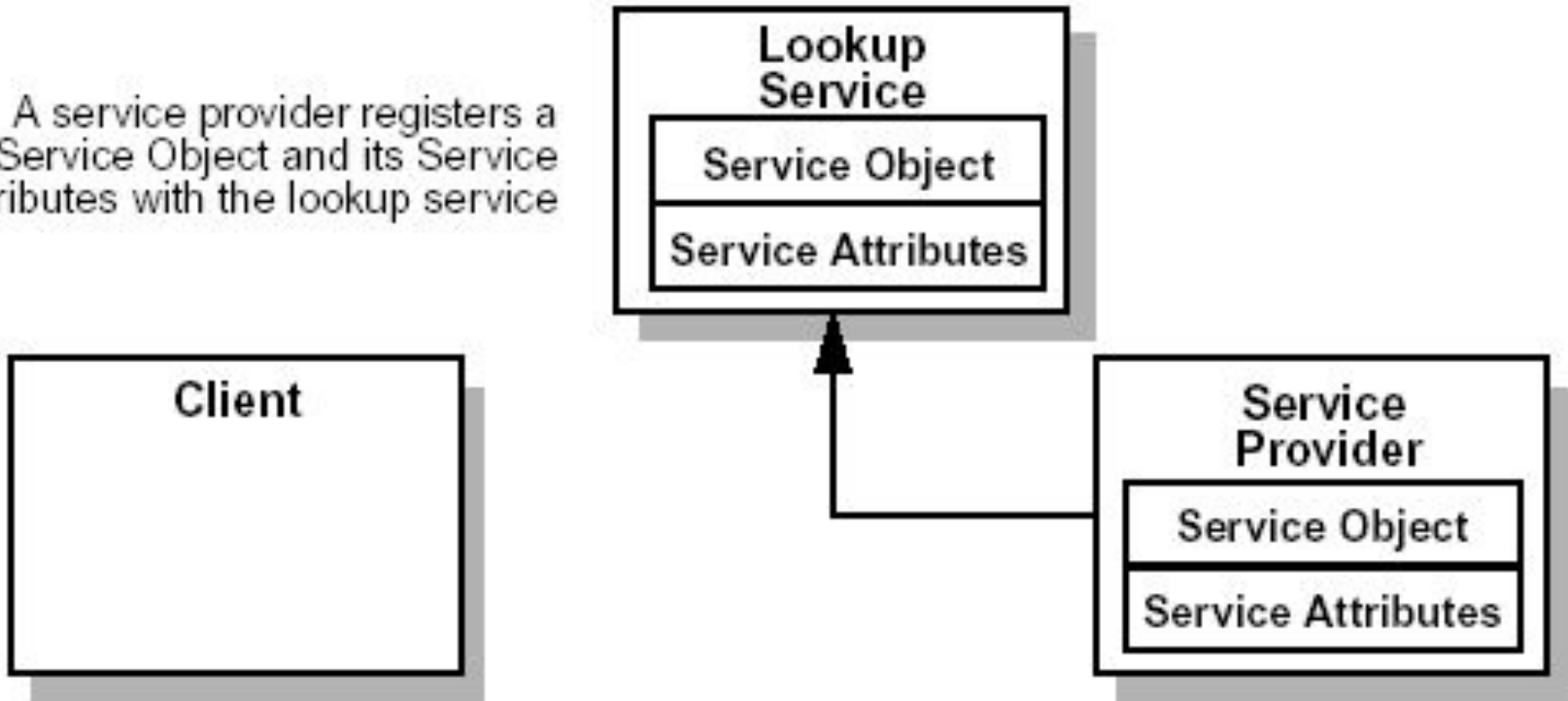
# Jini Service Discovery



# Jini

## Join & Lookup Protocols

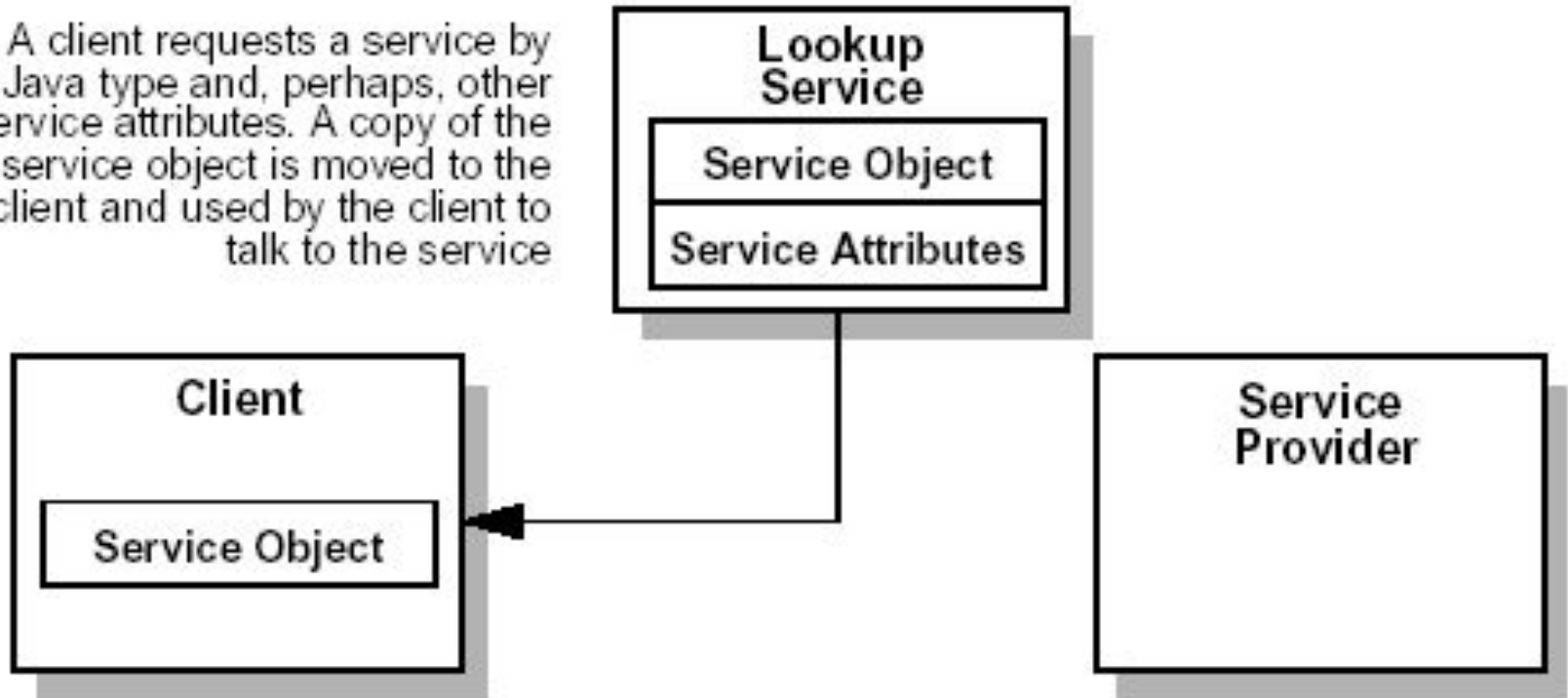
A service provider registers a Service Object and its Service Attributes with the lookup service



**Join**

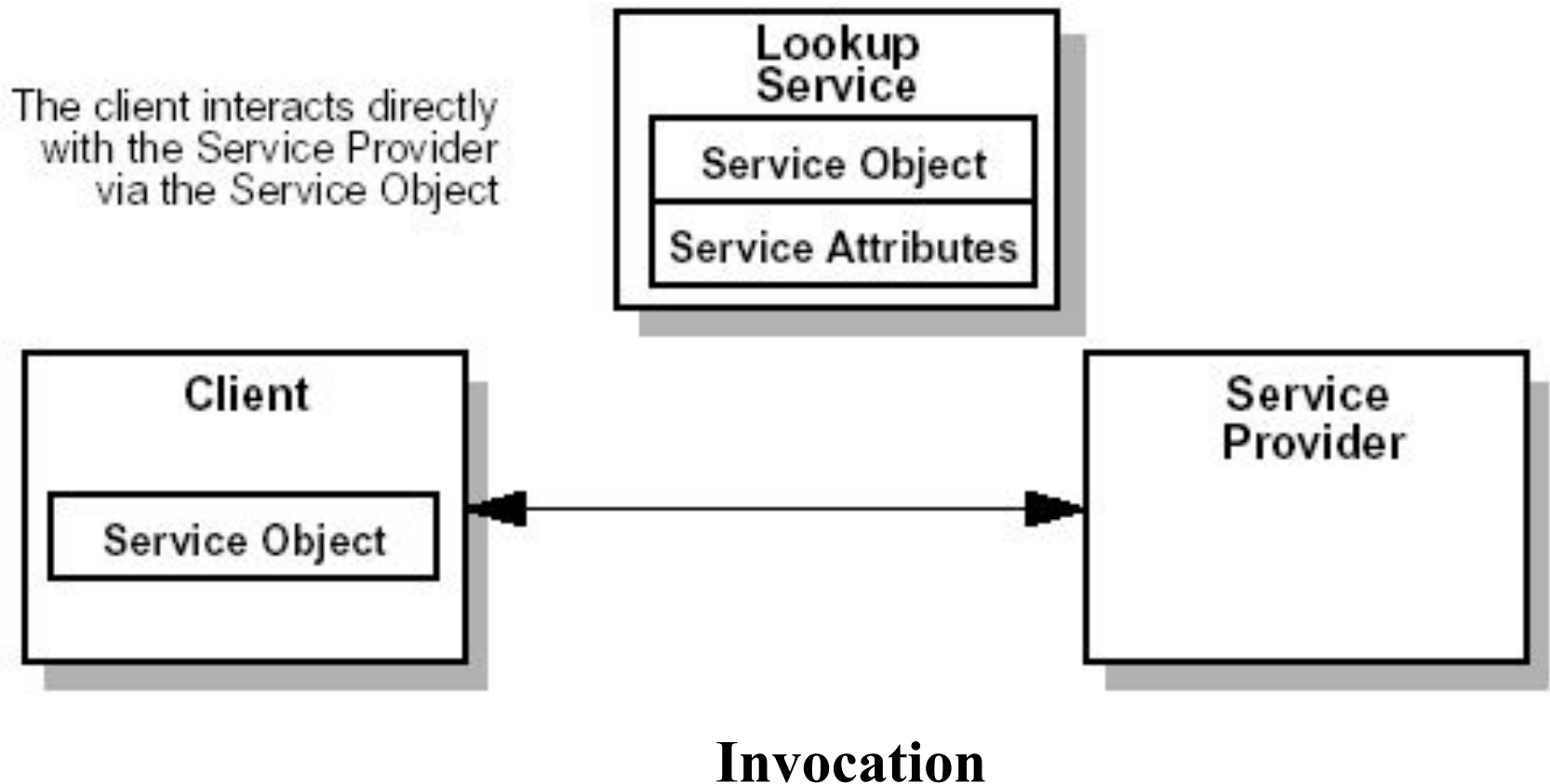
# Discovery, Join & Lookup Protocols

A client requests a service by Java type and, perhaps, other service attributes. A copy of the service object is moved to the client and used by the client to talk to the service



**Lookup**

# Discovery, Join & Lookup Protocols



# Attributes in Jini



- **Attributes describe service**
  - rich and flexible way for services to annotate their proxies with information describing that service
- **Attributes are Java objects**
  - assigned to service proxies
- **Attribute matching**
  - set of rules to determine when attributes match one another
- **template matching**
  - for matching against multiple attributes

# Discovery Protocol in Jini



- Serendipitous discovery
  - Jini allows serendipitous interactions between services and users of those services
  - Service initiated discovery
    - used when a service starts to find all lookup services in its vicinity
  - Lookup service initiated discovery
    - used when a lookup service starts and announces its presence to Jini services
- Hardwired (Direct discovery)
  - hardwire a Jini service to a lookup service

# Discovery Protocol



- Happens when a device first connects to the Jini System
  - Device could find/join multiple groups
- Unicast Discovery
  - For applications and services that know about particular lookup services.
- Multicast Discovery
  - Multicast Request - Device looking for Lookup Service in a group
  - Multicast Announce - Lookup Service Advertises its presence
  - Uses IP multicast based on UDP/IP
    - each message has a scope (distance) associated with it
    - promotes efficiency in routing
    - can set IP TTL (how many hops) parameter

# Join Protocol



- Registers a service with a Lookup Service in a Jini System
  - Each Service has a list of properties, Service ID, Attributes, a list of groups to register with, etc.
- Uses Discovery to find Lookup Services
  - Maintains a list of Lookup Services to register with
  - Registers with all Lookup Service that responds
  - Creates a lease during registration, which is renewed periodically

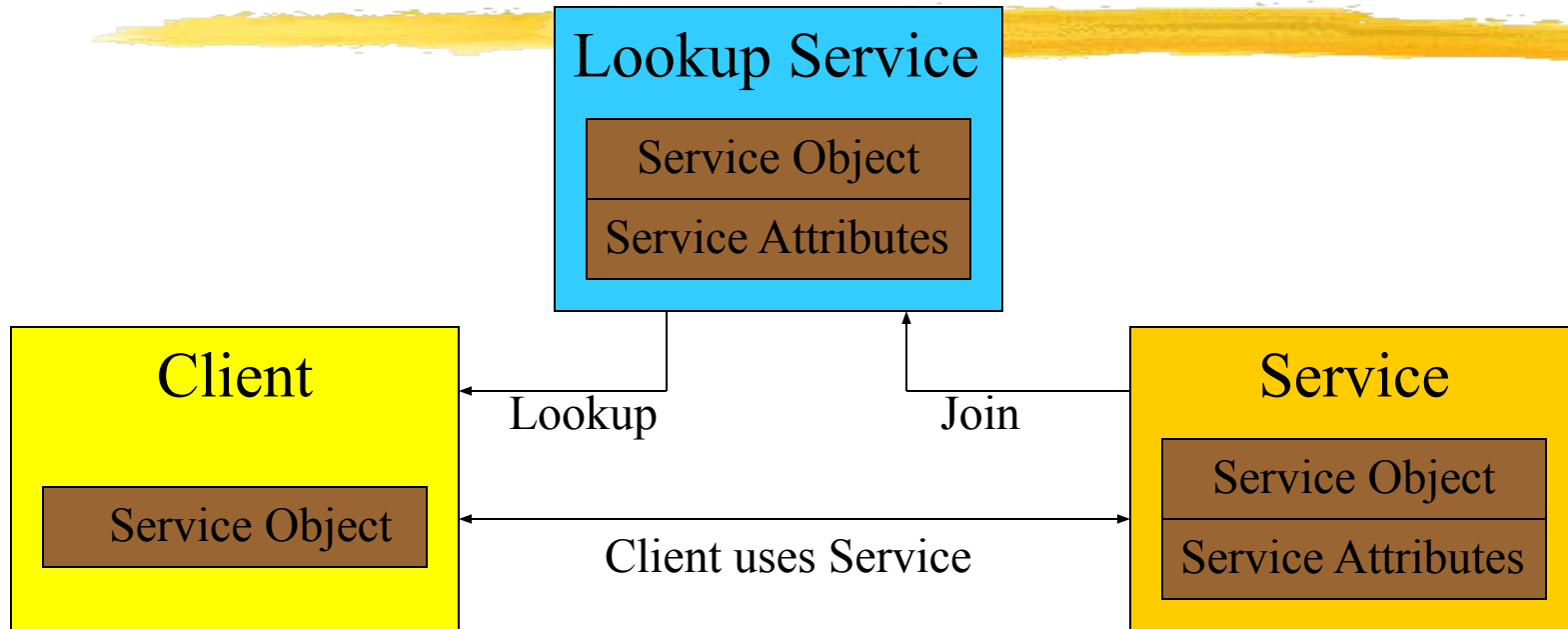


# Lookup Protocol



- Client queries the Lookup Service
  - Find a service by name, or attributes
  - Receive a copy of the service interface onto the client
- Client interacts with service through this “proxy” object
  - Client also gets a lease on the service

# Join and Lookup: An Example



- Join: Service object is registered.
  - Copy sent to reside on Lookup Service through RMI
- Lookup: Service is copied to Client
  - Service Object acts a proxy

# Service Architecture



- The service object on the client communicates with Service by:
  - RMI
  - Local implementation
  - Combination of the above (smart proxy)
- From client point of view:
  - Services look the same across the network or in local address space
  - All services are Java objects

# Security



- Based on principals and access control lists
  - Services accessed on behalf of some entity-the principal
    - Usually traces back to the user
  - Access is determined through an ACL associated with an object

# Programming Model



- The leasing interface
  - defines a way of allocating and freeing resources using a renewable, duration-based model
- The event and notification interface
  - an extension of the event model used by JavaBeans™ components to the distributed environment that enables event-based communication between Jini services
- Transaction interfaces
  - enable entities to cooperate in such a way that either all of the changes made to the group occur atomically or none of them occur
  - Jini provides an interface for two-phase commit transactions
    - Does not provide implementation
    - Does not define semantics of transactions
    - Only provides protocol to coordinate

# Leasing



- Set of interfaces that allow time-based resource allocation
  - Guarantees access to a service while lease is in effect
  - Can be renewed (depends in the service)
  - Can be exclusive or non-exclusive
  - Lease can be cancelled or it automatically expires at the end of the terms of the lease

# Jini Events



- Allows an object in one JVM to register for events occurring on another
  - Possibly across a network
  - Can register for different *kinds* of events
  - Can schedule notifications
- Provides interfaces that implement a protocol
  - No guarantees made interfaces, only by implementations

# Transactions in Jini



- Create a transaction
  - Jini transactionFactory object to create a transaction object to hold grouped operations
- pass to it all the transactions to be grouped
- tell to try to execute all operations atomically, which will either succeed or fail
  - commit() call



# Component Overview



	Infrastructure	Programming Model	Services
Base Java	Java VM RMI Java Security	Java APIs JavaBeans™ ...	JNDI Enterprise Beans JTS ...
Java + Jini	Discovery/Join Distributed Security Lookup	Leasing Transactions Events	Printing Transaction Manager JavaSpaces™ Service ...

# JINI summary



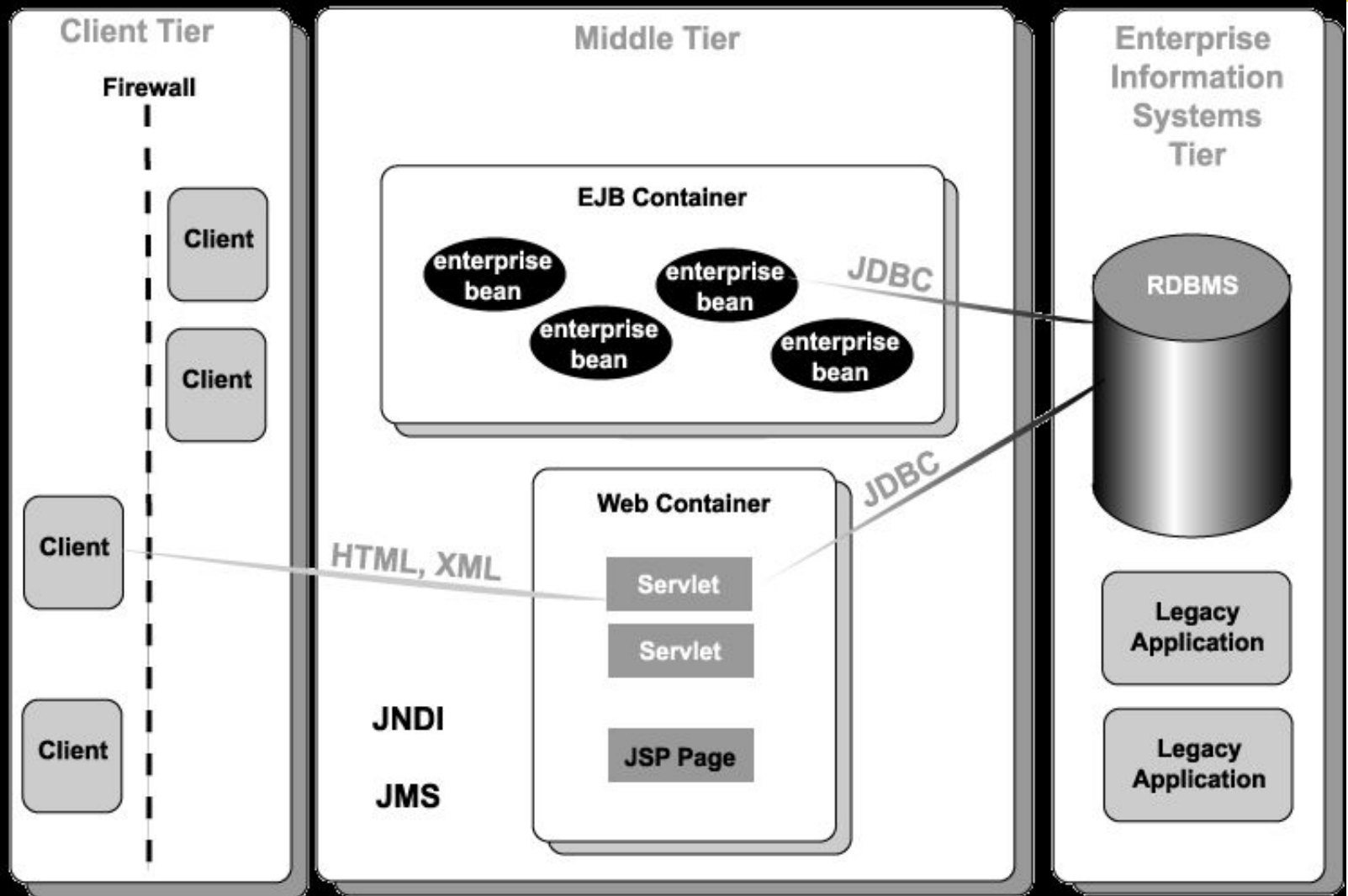
- Federate devices and software components into a single, dynamic distributed system
- ***Service:*** an entity that can be used by a person, a program, or another service
- ***Lookup Service:*** discovery, join, lookup
- ***RMI:*** Remote Method Invocation
- ***Security:*** principal, access control list
- ***Leasing:*** a grant of guaranteed access over a time period
- Transactions
- Events

# **Java-based Enterprise Platforms and Middleware**



**J2EE and EJB**

# J2EE Architecture



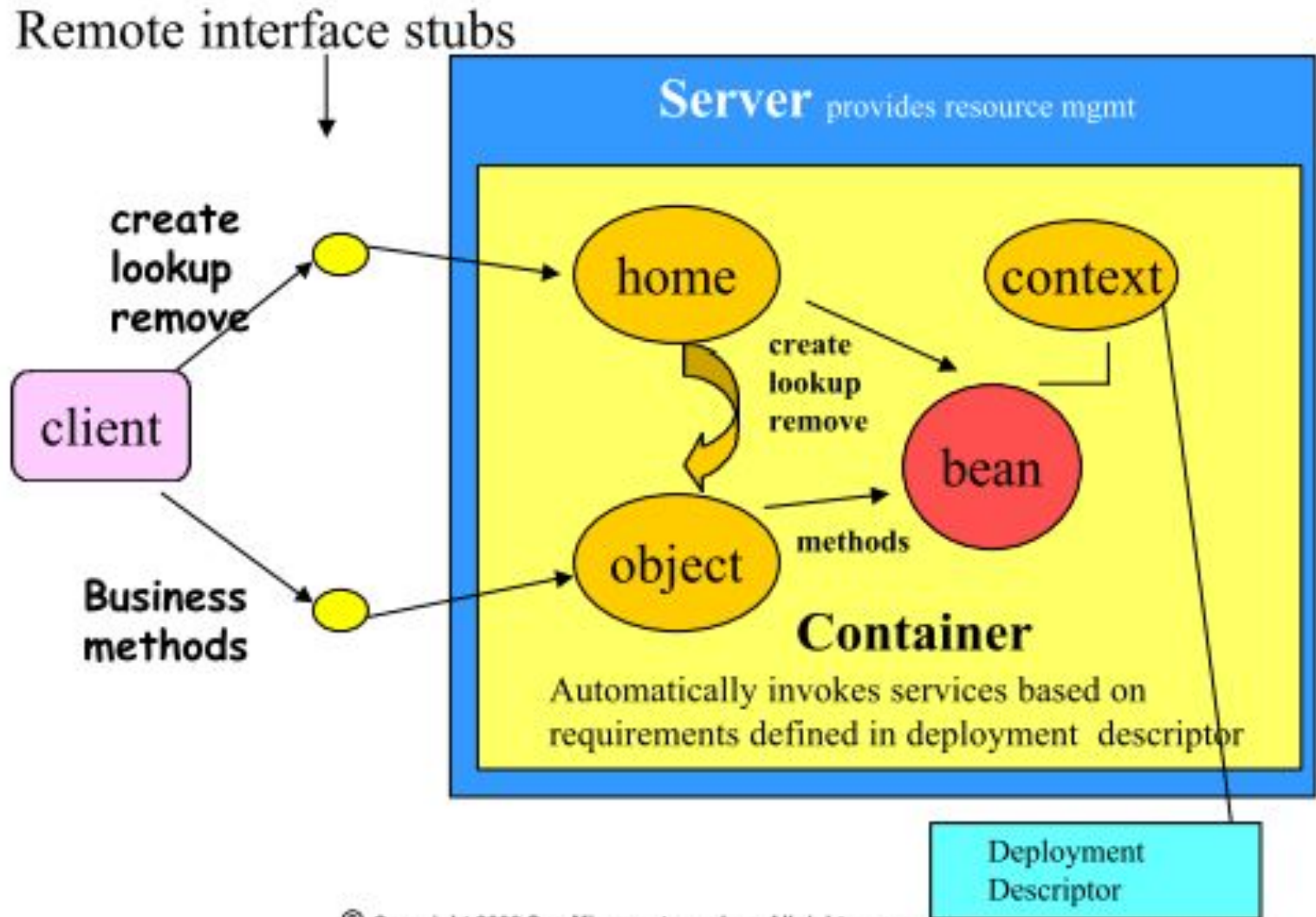
# Enterprise JavaBeans




A Server side distributed transaction component architecture (for J2EE)

- Encapsulates business logic and data in a container
- integrates directory services, configuration, security, transactions etc..
- Standard component model for application servers
- EJB enables rapid and simplified development of distributed, transactional, secure and portable Java applications.

# EJB Architecture



# Remote Interface



- WebAddressAccount.java
  - defines the business methods that a client may call. The business methods are implemented in the enterprise bean code

```
public interface WebAddressAccount extends EJBObject {
```

```
    public String getURLName();  
    public String getURLDescript();
```

```
}
```

# Home Interface



- WebAddressAccountHome.java
  - defines the methods that allow a client to create, find, or remove an enterprise bean
- ```
public interface WebAddressAccountHome extends EJBHome
{
```
- ```
    public WebAddressAccount create(String urlName, String
urlDescript);
```
- ```
    public WebAddressAccount findByPrimaryKey(String
urlName) ;
```
- ```
    }
```



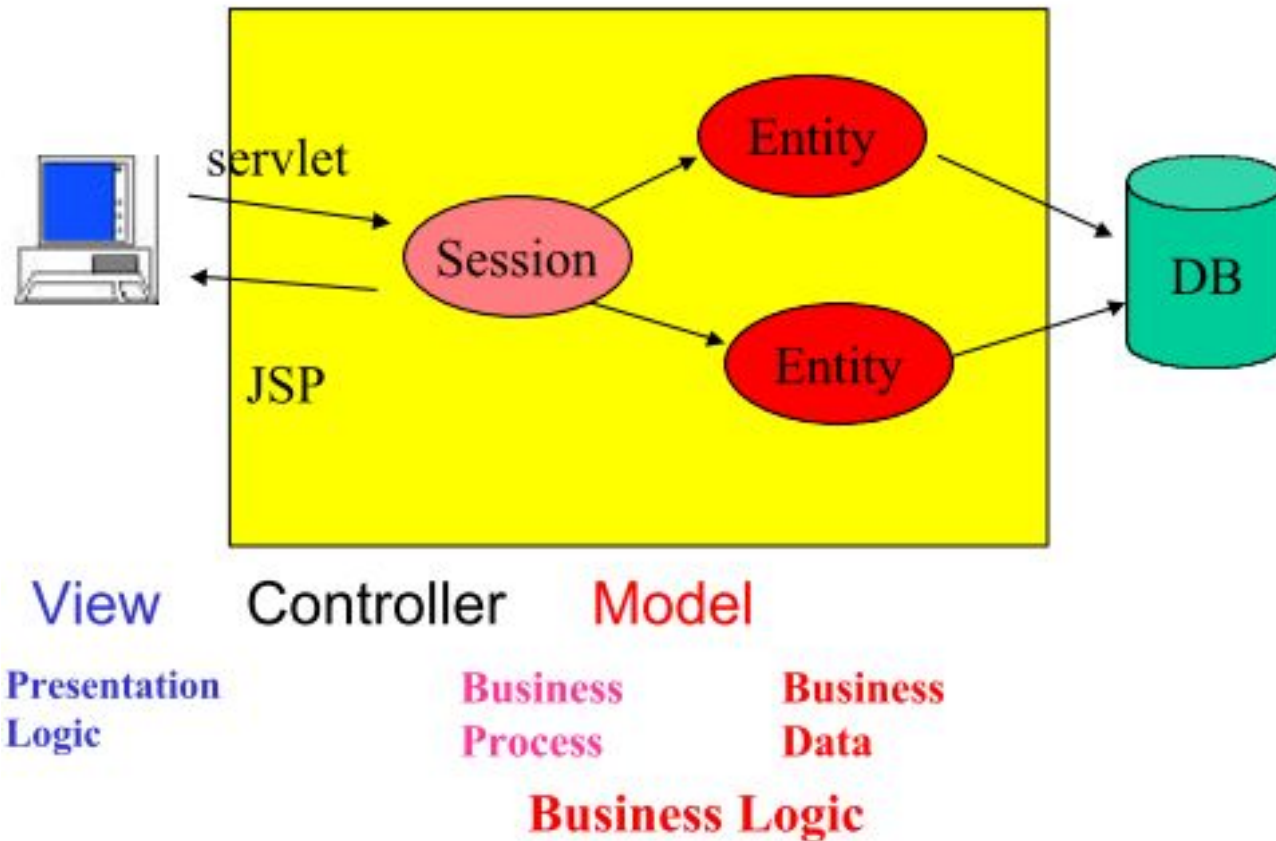
# Enterprise Bean Class



- WebAddressAccountBean.java
  - implements the business methods

```
public class WebAddressAccountBean implements EntityBean {  
  
    public String getUrlName() { return urlName;    }  
    public String getUrlDescript() { return urlDescript;    }  
    public String ejbCreate( String urlName, String urlDescript) {  
        insertRow( urlName, urlDescript);  
    }  
    public String ejbFindByPrimaryKey(String primaryKey) {  
        result = selectByPrimaryKey(primaryKey);  
    }  
}
```

# Thin Client Design Model



Later -- Message Driven Beans that talked to messaging platforms or backend databases;  
entity beans integrated into persistence architecture (JPA)

# Session Beans



- Represents business rules or process
- Perform work for individual clients on the server
- Encapsulate complex business logic
- Can coordinate transactional work on multiple entity beans
- 2 types: Stateful and Stateless
  - Stateful : session bean holds client state data

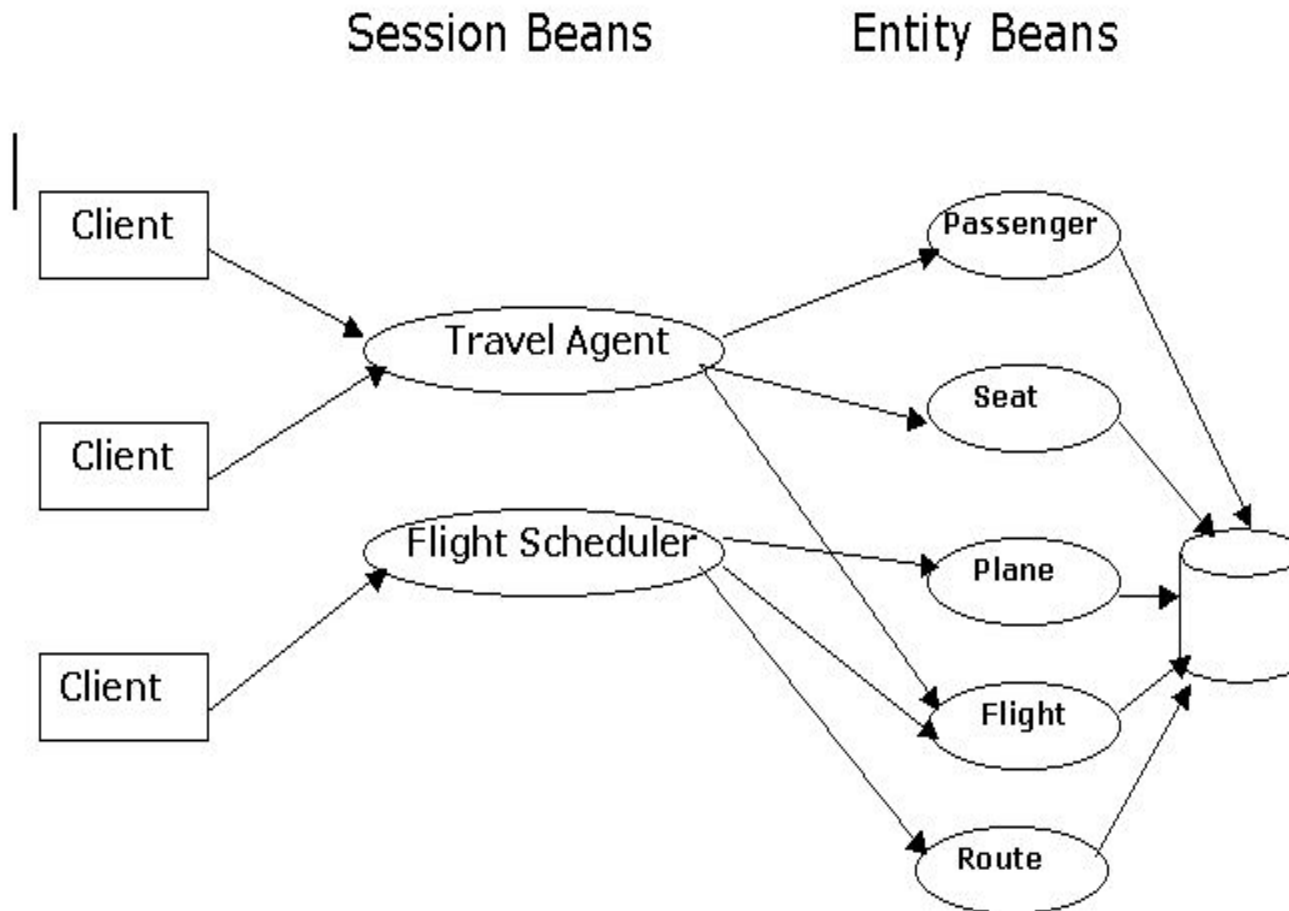
# Entity Beans



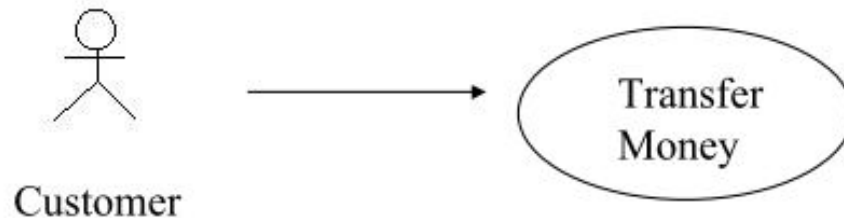
- Represents business model data
- ◎ Persisted in storage system ( usually Database)
- ◎ Might contain Application logic intrinsic to entity
- Maps business data to java class

# EJB Application Usecase

- Distributed Airline Travel Reservation System

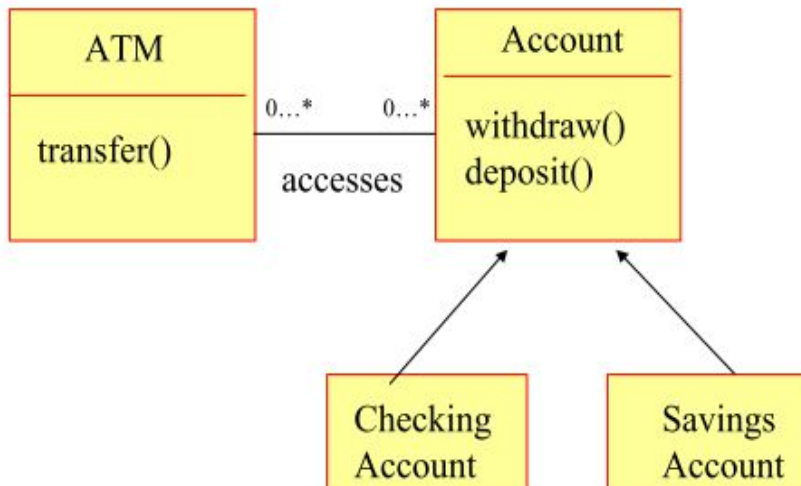


# EJB Use Case Banking System

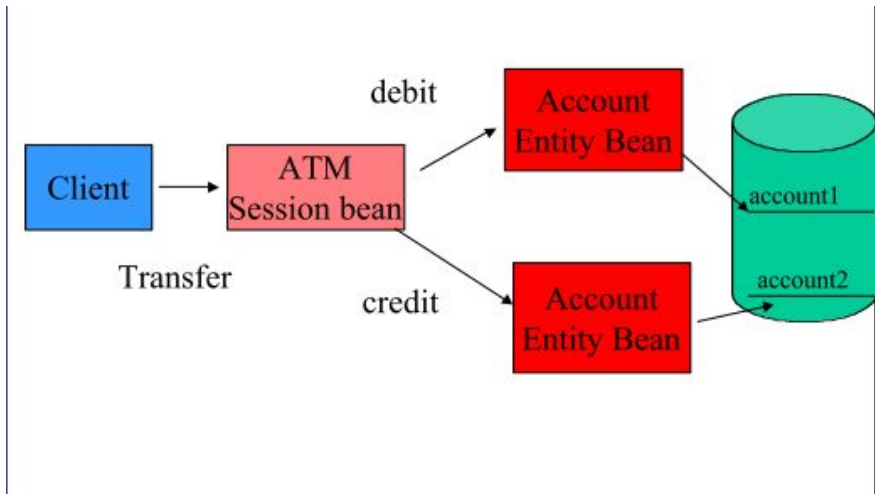


ATM Customer transfers money from checking to savings account

## Classes

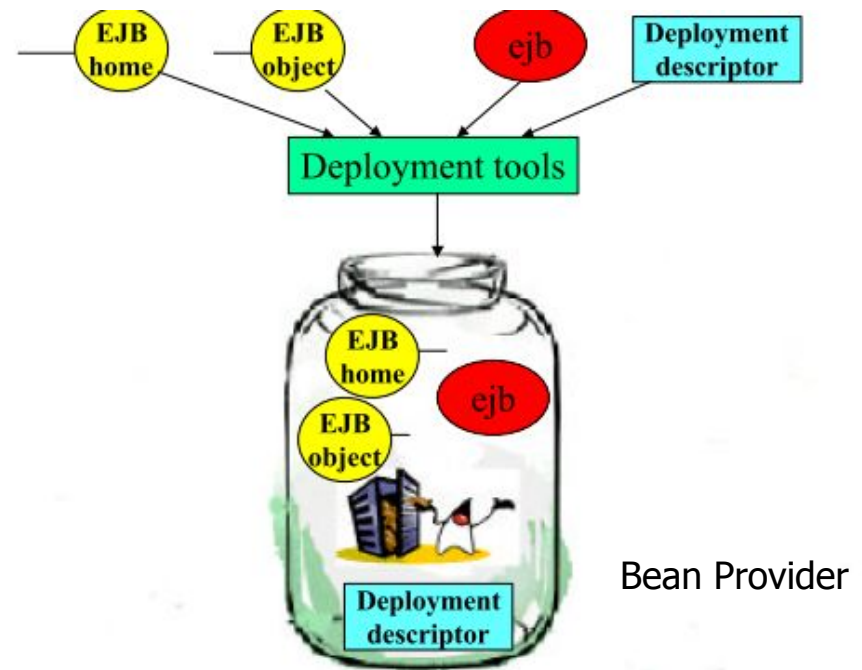
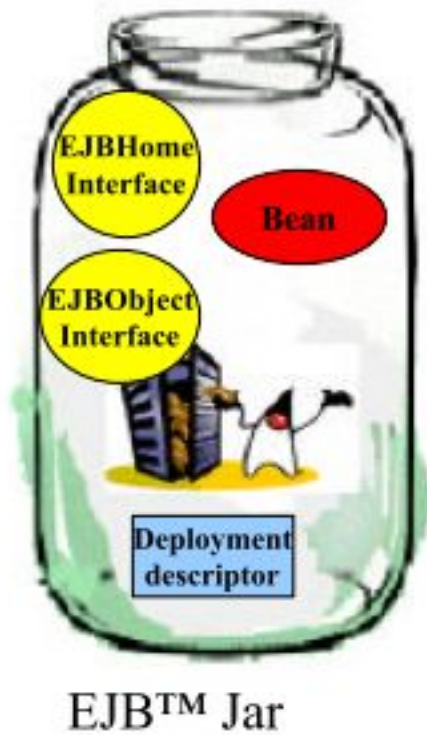


## EJB Representation

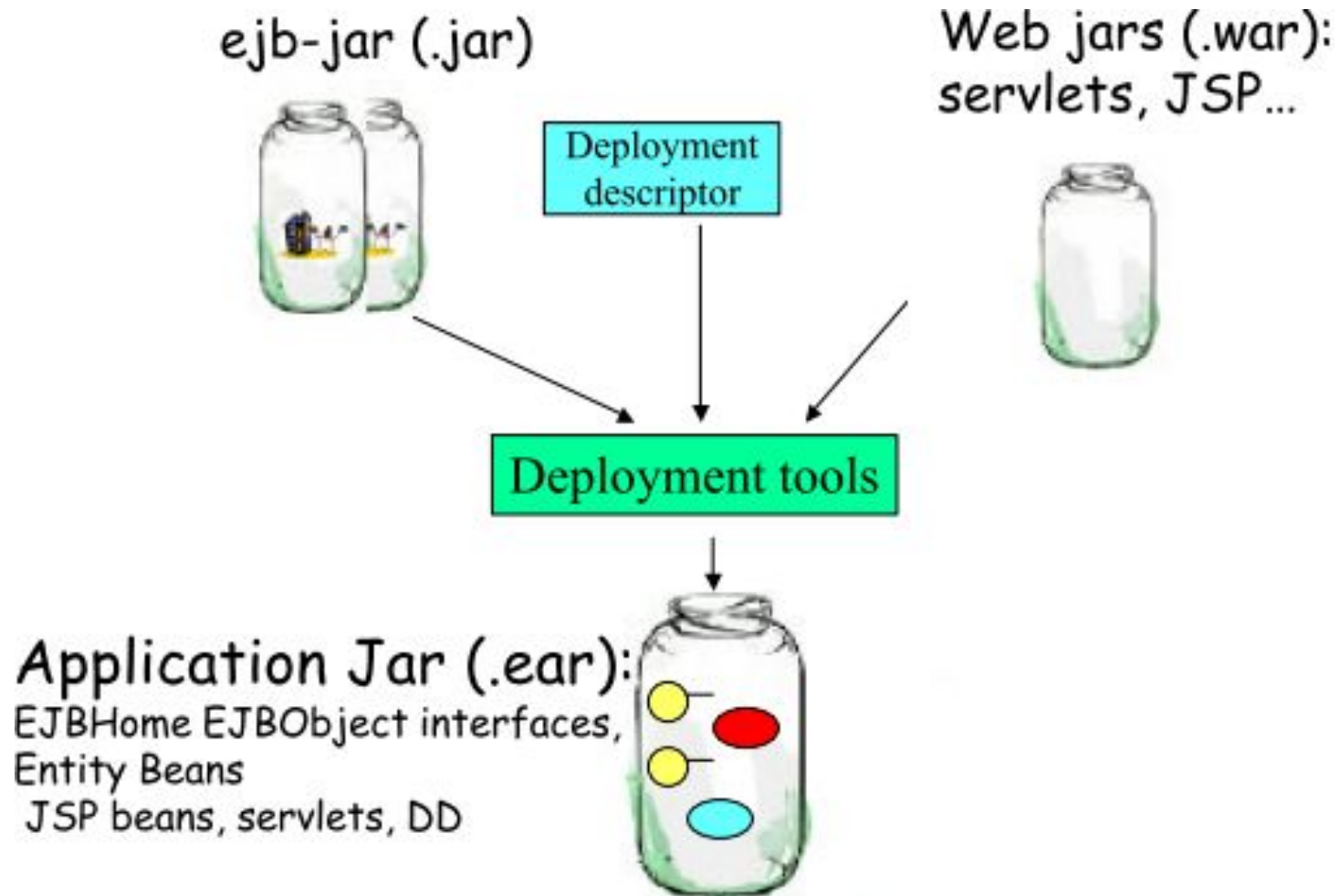


# EJB Packaging

- Packed in a jar file
- Factory
- Proxy
- XML Deployment Descriptor

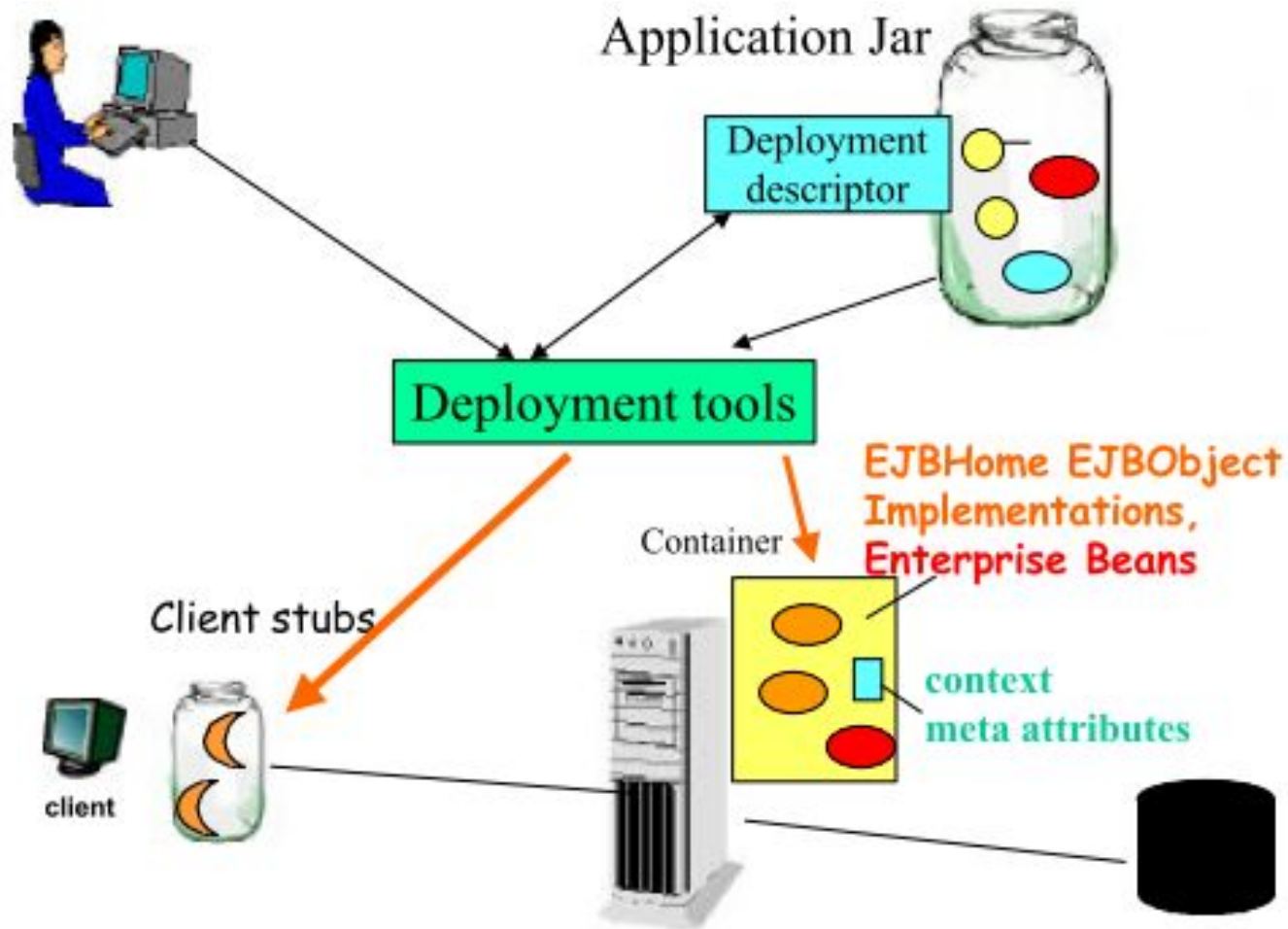


# Application Assembler

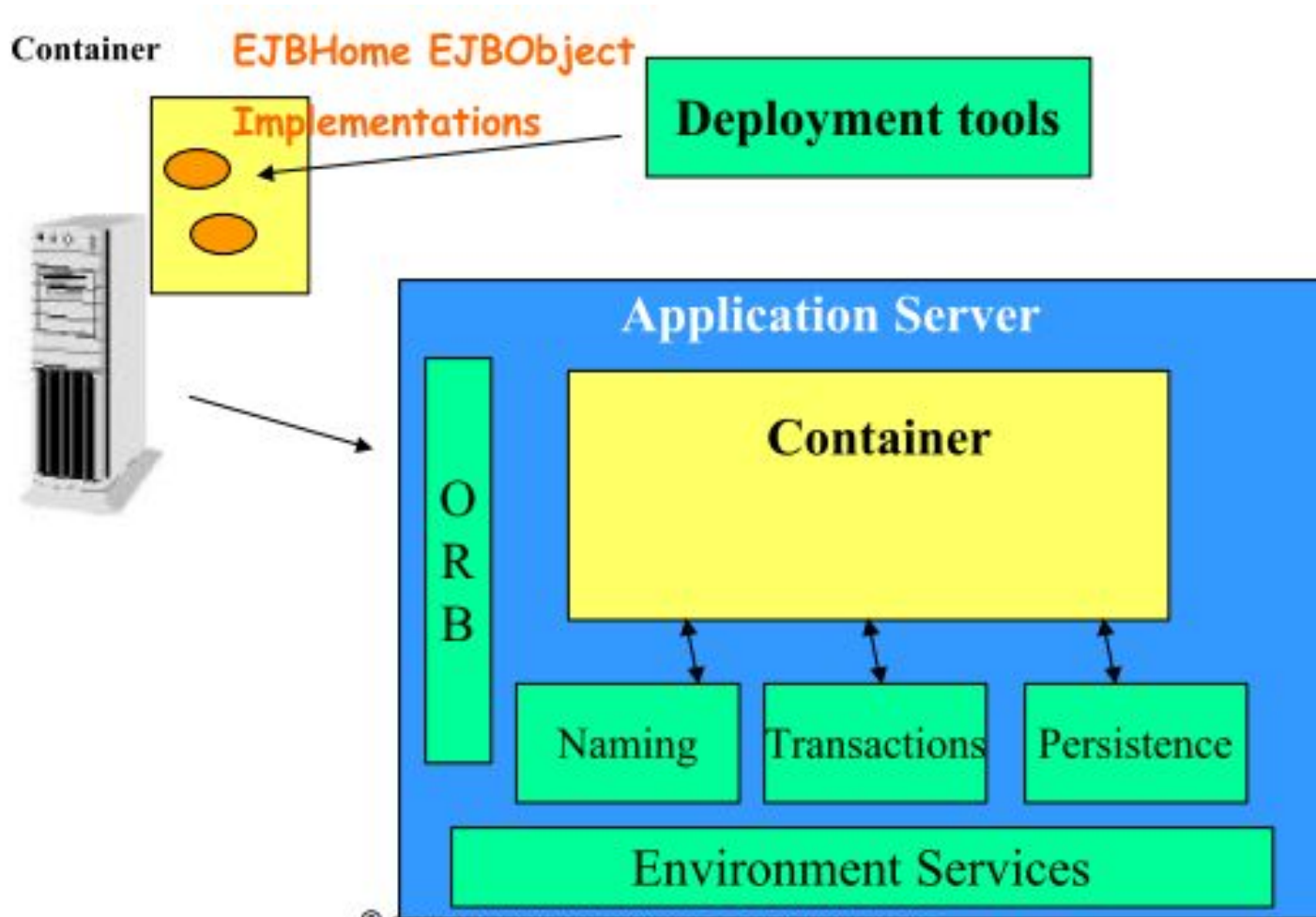




# Deployer



# Service/ Container Provider



# Travel Reservation System: Bean Provider

---

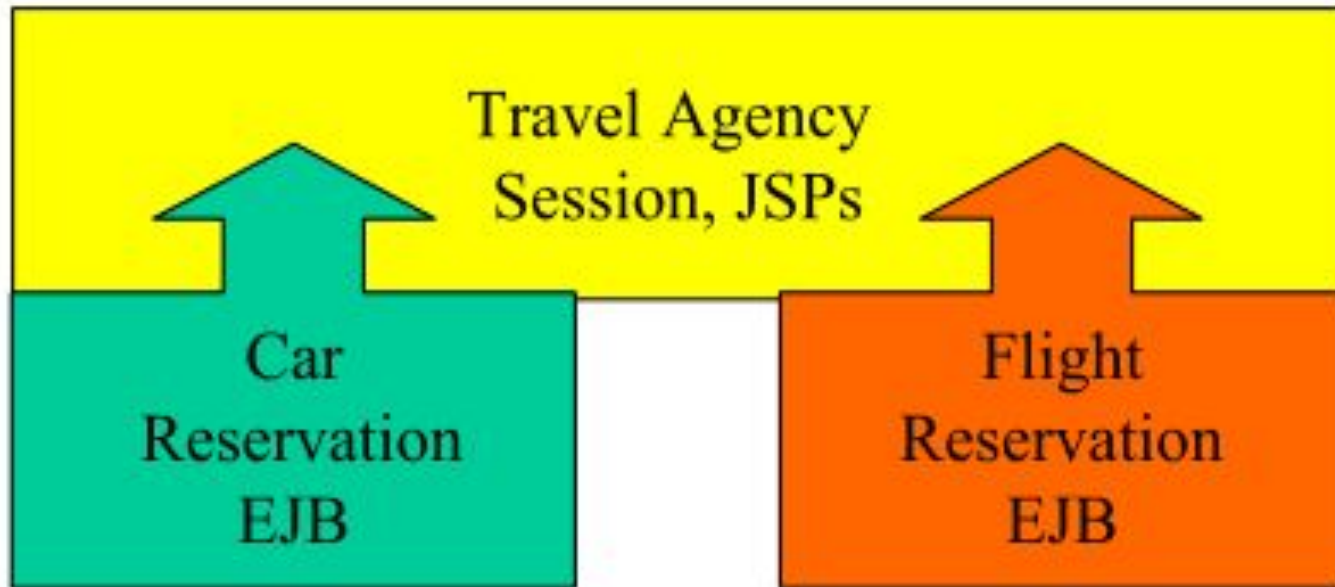
Hertz Bean Provider  
Creates Car Reservation EJB(s)



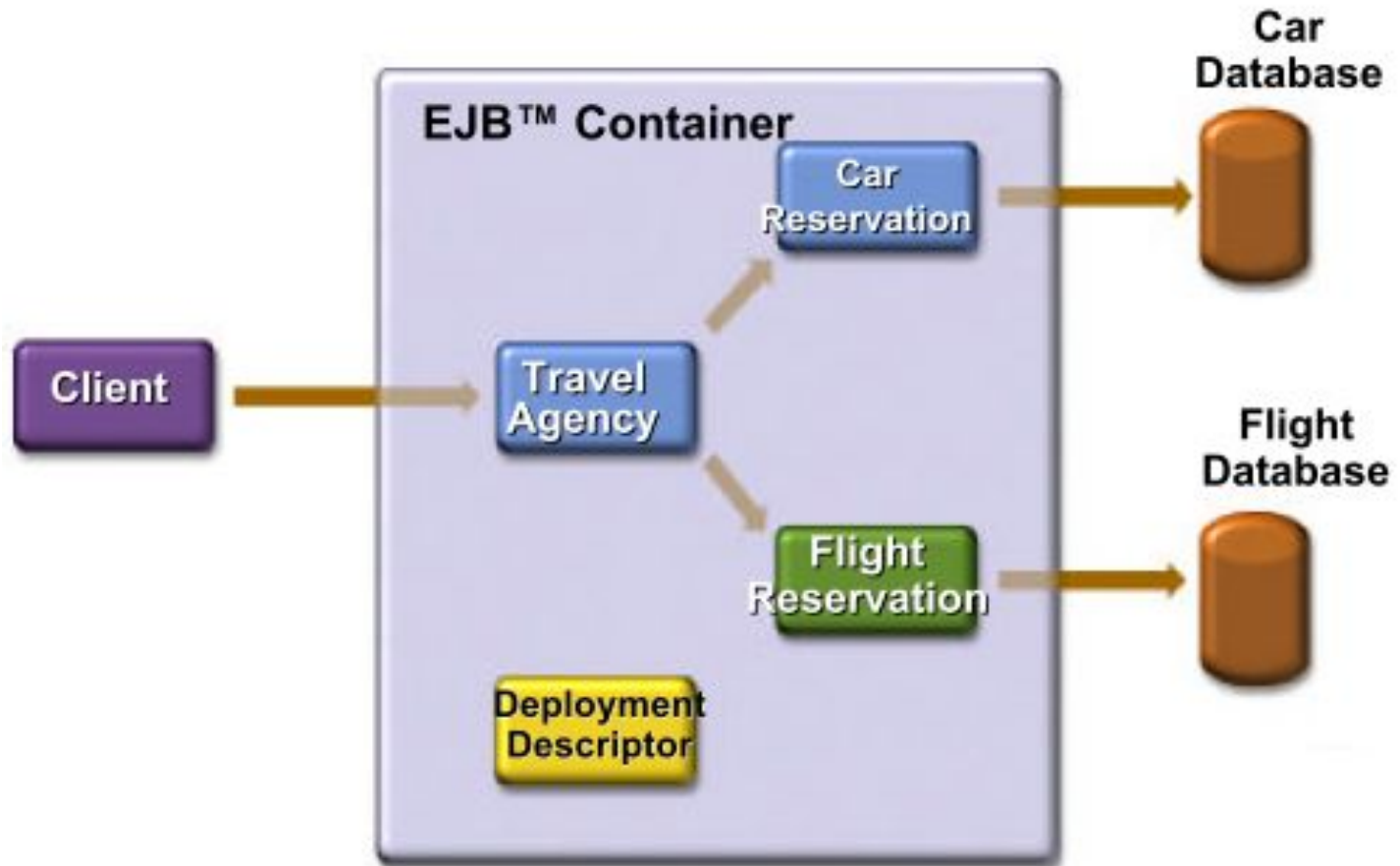
Delta Bean Provider  
Creates flight Reservation EJB(s)



# Application Assembler



# Deployment in a target container



# Features



- Portable
- Contained and Managed at Runtime
- Simplifies the complexity of building n-tier application
- Scalable & distributable
- Easy to upgrade and maintain

# J2EE Motivation



- ⦿ New multi-tier enterprise computing model in web environment
- A way to bring in different elements of enterprise application:
  - Web interface design
  - Transaction processing
  - Meeting non-functional system requirements:
    - Availability, reliability, extensibility, performance, scalability, reusability, interoperability
  - Timely development and deployment

# Java Based Enterprise Platforms



- Platform introduced - 1999
- J2SE – Java 2 Standard Edition
  - Java for the desktop / workstation
  - <http://java.sun.com/j2se>
- J2ME – Java 2 Micro Edition
  - Java for the consumer device
  - <http://java.sun.com/j2me>
- J2EE - Java 2 Enterprise Edition
  - Java for the server
  - <http://java.sun.com/j2ee>

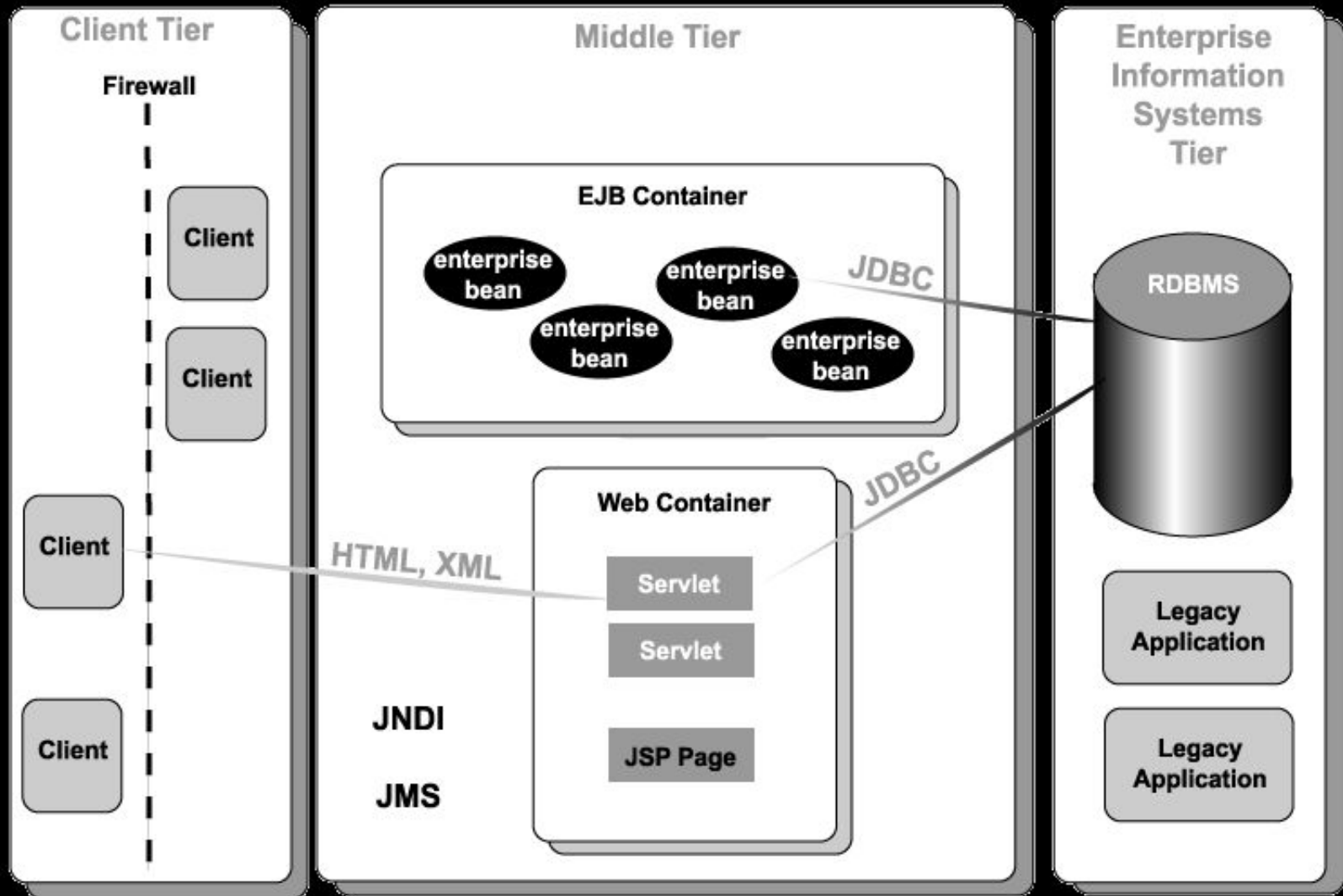


# What is J2EE?



- A Multi-tiered distributed application model
- A collection of Standards: JDBC, JNDI, JMX, JMS
- A Component Technology: EJB
- An Application Server

# J2EE Architecture

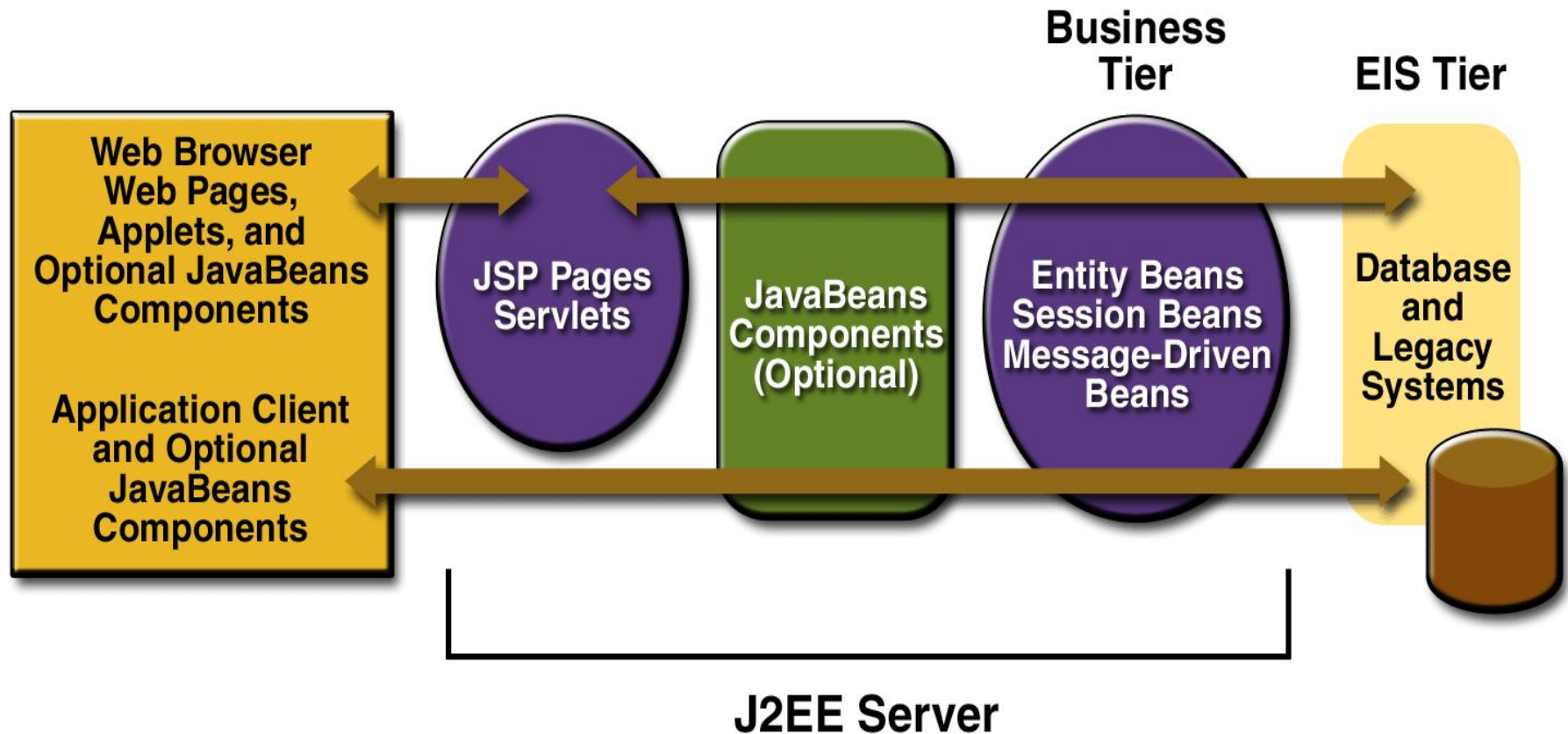


# J2EE Tiers



- Client Presentation
  - HTML or Java applets deployed in Browser
  - XML documentations transmitted through HTTP
  - Java clients running in Client Java Virtual Machine (JVM)
- Presentation Logic
  - Servlets or JavaServer Pages running in web server
- Application Logic
  - Enterprise JavaBeans running in Server

# J2EE Tiers



# J2EE Components and Services



- **Components**
  - Java Servlets
  - JavaServer Pages (JSP)
  - Enterprise JavaBeans (EJB)
- **Standard services & supporting technologies**
  - Java database connectivity(JDBC) data access API
  - Java Messaging Service (JMS)
  - (Remote Method Invocations (RMI))
  - Extensible Markup Languages(XML)
  - JavaIDL (Interface Description Language)
  - JavaMail
  - Java Security
  - CORBA technology
  - ***Design Patterns***

# J2EE Clients



- Web Clients (thin clients): dynamic web pages and a web browser
- Applets: Client application in Java that runs on JVM on the web browser
- Application Clients: Runs on a client machine to provide a way for users to handle tasks that require a richer user interface

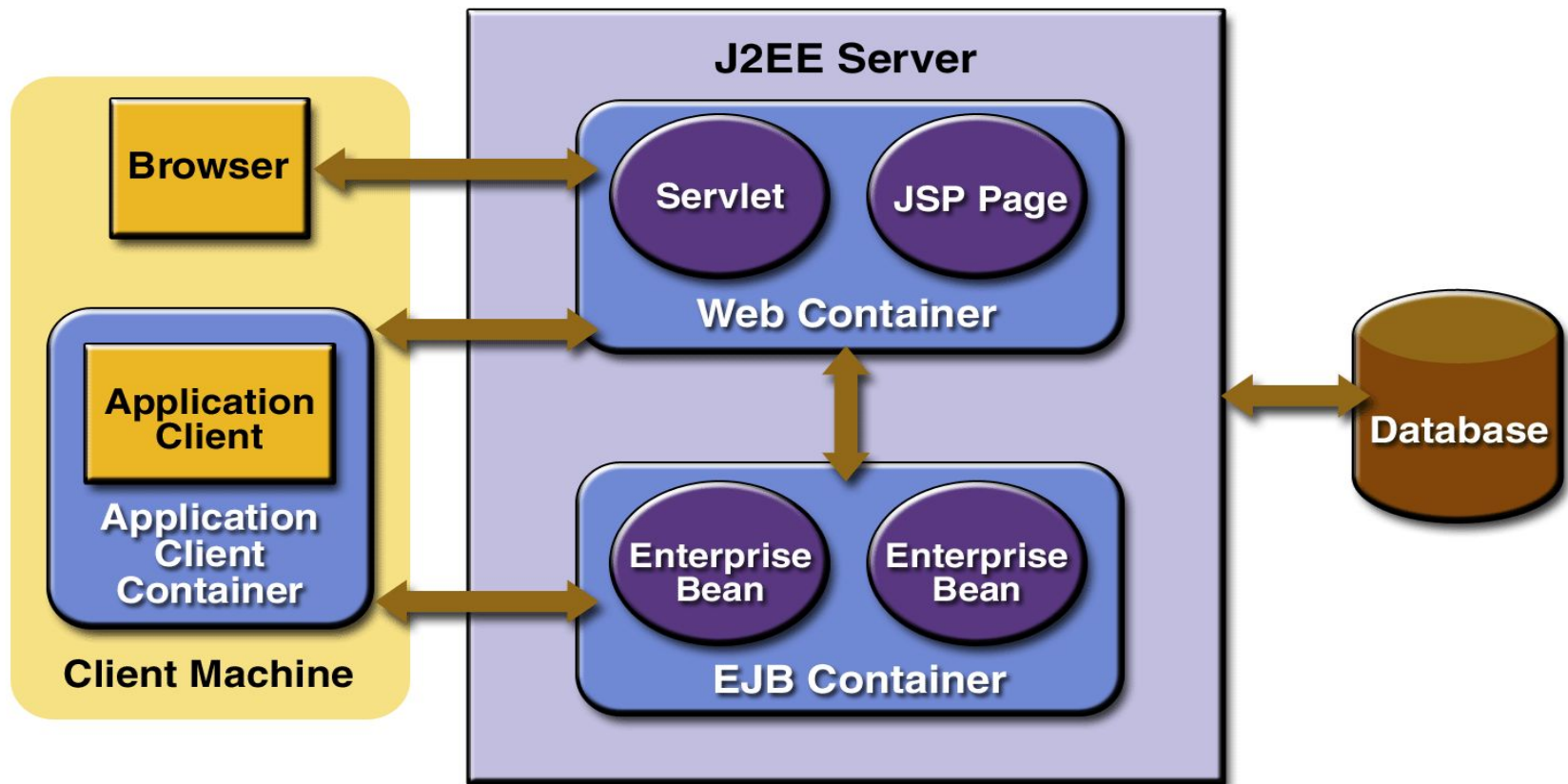
# Enterprise Information System Tier



- Information Infrastructure for an enterprise
- Handles enterprise information system software and includes enterprise infrastructure systems such as enterprise resource planning (ERP)
- Necessary to ensure transactional access to EIS system from various applications

# J2EE Containers

An interface between a component and a low-level platform specific functionality





# J2EE APIs



- Enterprise JavaBeans Technology 2.0
- JDBC API 2.0
- Java Servlet Technology 2.3
- Java Server Pages Technology 1.2
- Java Message Service 1.0
- Java Naming and Directory Interface 1.2
- Java Transaction API 1.0
- Java Mail API 1.2
- Java API for XML Processing 1.1
- Java Authentication & Authorization Service 1.0

# What is Java Servlet?



- Conforms to Java Servlet API in J2EE
- Container managed Web Component
- Generate dynamic response to requests from web based clients
- Synchronize multiple concurrent client request

# What is Java Server Pages?



- Conforms to J2EE Web Application
- Web Component that sits on top of Java Servlet mode
- Dynamically generates Web pages based on HTML, XML
- Text based documents describe how to process a request and create a response

# References



- <http://www.scribd.com/doc/6173018/J2EE-Tutorial>
- [http://java.sun.com/j2ee/tutorial/1\\_3-fcs/doc/Overview6.html](http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Overview6.html)
- [http://download.oracle.com/docs/cd/B31017\\_01/migrate.1013/b25219/overview.htm](http://download.oracle.com/docs/cd/B31017_01/migrate.1013/b25219/overview.htm)
- <http://en.wikipedia.org/wiki/J2ee>
- [http://rangiroa.essi.fr/cours/ejb/00-ejbcodecamp\\_ch1of6.PDF](http://rangiroa.essi.fr/cours/ejb/00-ejbcodecamp_ch1of6.PDF)

