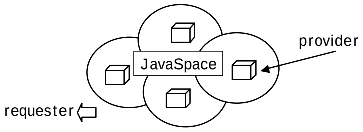
JavaSpaces and Jini

JavaSpaces

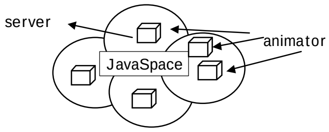
JavaSpaces

- Distributed systems model of computation
- NOT client-server
- Requesters and providers of services are decoupled



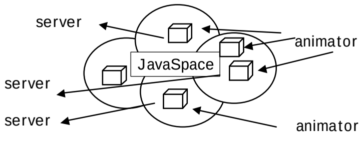
Example

- Animator needs to render frames for a movie
- Writes “request for rendering” entries
- Servers pick up requests – do the work
- Return results to JavaSpace



Scale Up

- Add more servers
- Add more animators



Entries

- Participants communicate by exchanging entries in the space
- An entry is a types group of java objects (Entry is a java class)
- Entries are put into a space using the write method
- Participants examine the space using read and take

Entries, Templates, and Operations

Four basic operations on a space – params are Entries and Templates (a special kind of entry)

- write
  - writes Entry to space
- read
  - read Entry that matches a template
- take
  - same as read but removes entry
- notify
  - notify an object when entry matching template arrives in the space

Entry

- An interface
- public interface Entry extends Serializable { . . . }
  - no code required

Entry with one FIELD

```
import net.jini.space.Entry;
pubilc class MyEntry implements Entry
{
  public String content;
  public MyEntry() { }
  public int foo() { // do something }
}
```

public fields
- objects only

Use RMI to get a JavaSpace

```
RefHolder rh =
  (RefHolder) Naming.lookup(“JavaSpace”);

// use the RefHolder’s proxy method to get the space
  JavaSpace jspace = (JavaSpace)rh.proxy();
```

Write Entry to Space

```
//create an Entry to write in to the space
MyEntry msg = new MyEntry();
msg.content = “ How do you do?”;
//The transaction under which to perform the write
Transaction txn = null;
//The lease duration for this entry
long timeToLive = Lease.FOREVER;
jspace.write(msg, txn, timeToLive);
```

Read Entry from Space

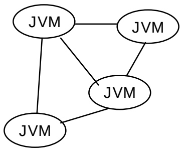
```
MyEntry template = new MyEntry();
//Set attribute to be null, act as wildcard

template.content = null;

//time to wait and transaction
long timeToWait = 0L;
Transaction sotxn = null;

MyEntry result = (MyEntry)jspace.read(template, sotxn,
                                              timeToWait);
System.out.println(result.content);
```

JavaSpaces Matching Process

- Default:
  - match on class or subclass
  - match on field value is specified
  - if template value is null, any value matches
- User defined matching
  - use of equals() method
  - equals(Entry, Entry)
  - allows return of different type

Jini

Jini – A Network of Java Virtual Machines



code and data can move from machine to machine as needed

JVMs can join and detach from the network

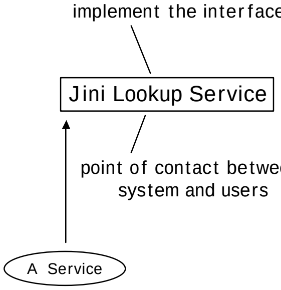
Jini Services

- An entity that can be used by a
  - person
  - program
  - another service
- A service can be
  - a computation
  - storage
  - communication channel
  - filter
  - hardware device

Example Services

- Printing a document
- Translating from one language to another

  - Services can be collected and assembled
  - Don't think client-server



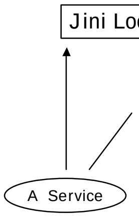
maps interfaces to objects that implement the interfaces

Jini Lookup Service

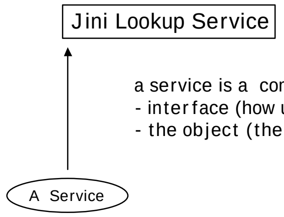
point of contact between system and users

A Service

Discovery / Join



a service is added to a lookup service by a pair of protocols:
-discovery (find a lookup service)
- join (join it)



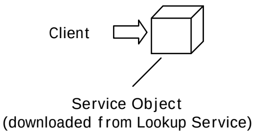
a service is a combination of:
- interface (how users can access)
- the object (the interface implementation)

Discovery / Join Protocol

Lookup Protocol

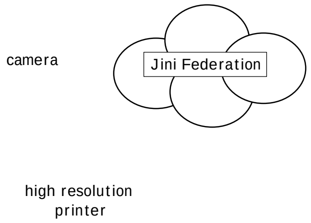
- A client location finds a service by its interface (TYPE)
  - also possibly other descriptive attributes
- The service object is loaded into the client (via RMI)
  - the client "talks" to the interface

Client and Service Provider

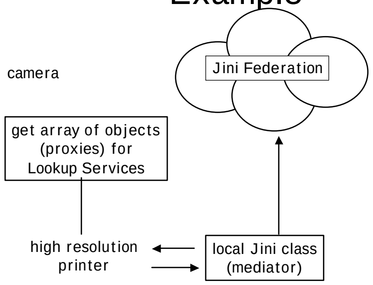


May be:
-Proxy object
-Actual object
-Some combination

Example



Example



camera

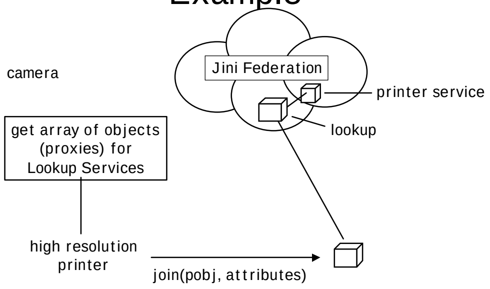
Jini Federation

get array of objects (proxies) for Lookup Services

high resolution printer

local Jini class (mediator)

Example



camera

Jini Federation

printer service

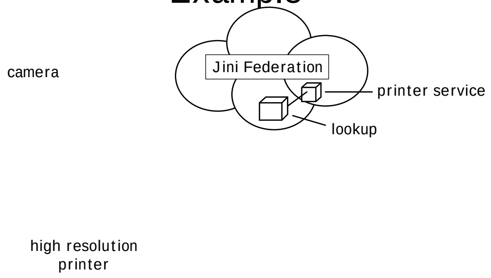
lookup

get array of objects (proxies) for Lookup Services

high resolution printer

join(pobj, attributes)

Example



camera

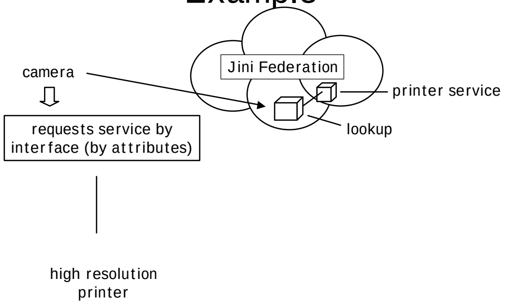
Jini Federation

printer service

lookup

high resolution printer

Example



camera

Jini Federation

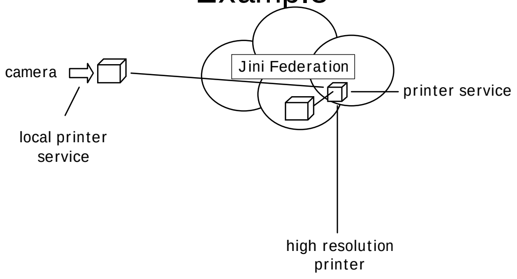
printer service

lookup

requests service by interface (by attributes)

high resolution printer

Example



camera

Jini Federation

printer service

local printer service

high resolution printer

Links

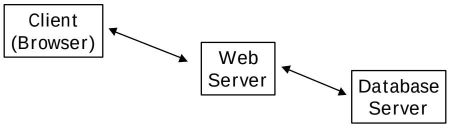
- http://java.sun.com/products/javaspaces
- http://www.sun.com/jini/

The Distributed Landscape

How do they fit together?

- Java Beans
- Enterprise Java Beans
- Java Server Pages (JSP)
- XML
- Jini and JavaSpaces

Classic 3 Tiers (Web-Enabled)

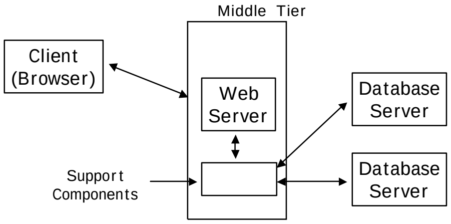


Client (Browser)

Web Server

Database Server

N Tiers



Middle Tier

Client (Browser)

Web Server

Support Components

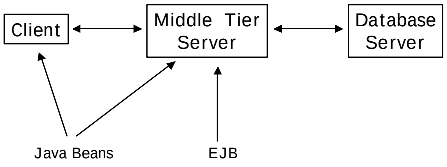
Database Server

Database Server

Enterprise Java Beans (EJB)

Enterprise Java Beans

- Component architecture for Java
- EJB is an API
  - vendors provide implementation
- EJB based on other Java APIs
  - RMI, CORBA, IIOP, JTS



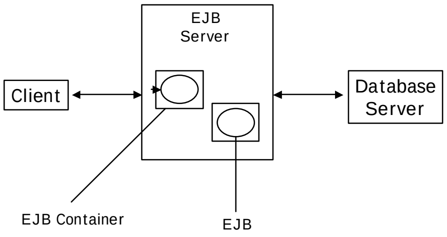
EJB (Server Components)

- Simplifies the development of complex ENTERPRISE applications
- Components contain only business logic
- Pluggable, reusable components
- Scalability
- Resource Management
- Transaction support
- Concurrency management

EJB Roles

- Component Developer
  - "Order entry Bean"
- Server/Container provider
  - 3^{rd} party vendors (e.g., BEA)
- Deployer
  - sets security parameters
- Tool Vendor
- Application Developer
  - connects beans

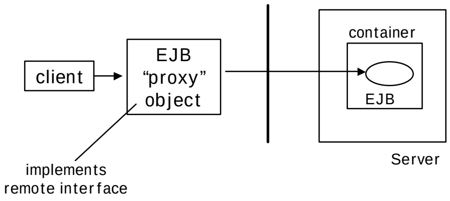
EJB Server provided by vendor –
must implement the EJB spec



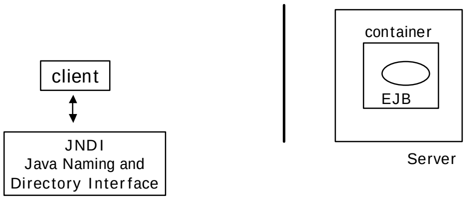
EJB Architecture

- Enterprise Java Bean
  - a Java class
  - implements some business logic
- EJB Container
  - a class that manages the Bean
  - if a Bean decides that it cannot complete its job as part of a transaction, it notifies its container which handles the details.

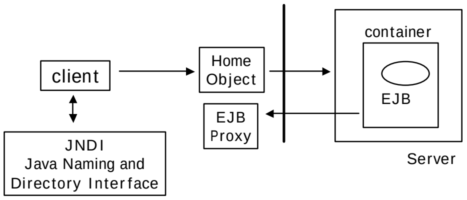
Clients talks to "proxy" object



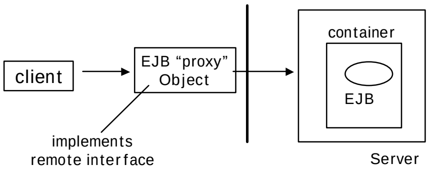
How does client get a “proxy” object ?



Client gets a “Home Interface” object – to create the “proxy”



Client talks to “proxy” object



Types of EJB

- Session Beans
  - live for the lifetime of a session
  - each client gets a copy
  - state not saved
- Entity beans
  - represents information stored persistently in a database
  - e.g., Account
  - state is preserved

EJB Package (in a jar file)

- The EJB – the Java class file(s)
- Remote Interface
  - the methods client will call
- Home Interface
  - methods to create (session), find (entity) and destroy Beans
- Deployment Descriptor
  - creation/persistence/transaction/security settings

Container

- Typically created by EJB implementation
- Uses Deployment Descriptor of EJB to:
  - return proxy object to client
  - insert transaction logic into calls to components
  - do security: control access to the Bean's services

Container Responsibilities

- Intercept request from client
- Lifecycle duties
  - automatic start component or thread when request arrives
- Persistence duties
  - automatically save the component’s state to disk (for entity beans)
- Transaction duties
  - based on Deployment Descriptor of Bean

Six Transaction Options of EJB

(Specified in Deployment Descriptor)

- BEAN_MANAGED
  - EJB itself handles T logic
- NOT_SUPPORTED
  - EJB cannot run in a T
- SUPPORTS
  - will run in T if one is active
  - will run outside T if none is active

Transaction Options (cont’d)

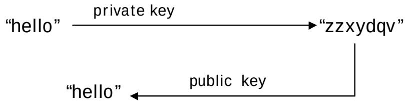
- REQUIRES
  - must have a T
  - container starts a T if none active
- REQUIRES_NEW
  - container starts new T on every call
- MANDATORY
  - if a T not active, an exception is thrown

EJB Security

- Container defines the security options
- Digital certificates automatically issued for Authorization
  - guarantees that the sender really sent the document
  - guarantees that the document received really is the document sent

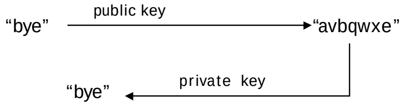
Public and Private Key Cryptography

- Two complementary keys are created at the same time using a mathematical formula

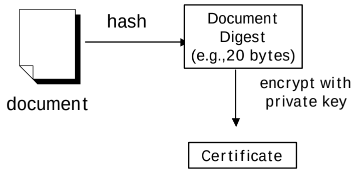


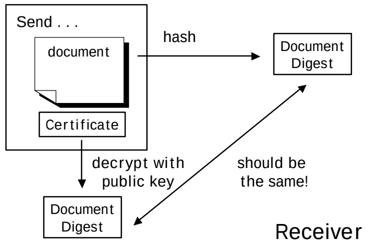
Public and Private Key Cryptography

- Two complementary keys are created at the same time using a mathematical formula



Digital Certificates



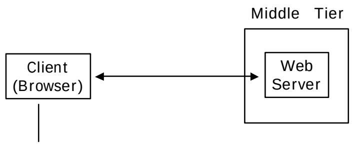


EJB Summary

- EJB takes the component model to the application level
- Provides automatic support for complex software
  - transactions
  - security

Java Server Pages (JSP)

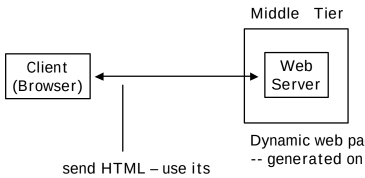
Client Browser



1. understands HTML
2. can run Applets
3. can trigger programs on server

BUT, Applets are dependent on browsers doing Java right

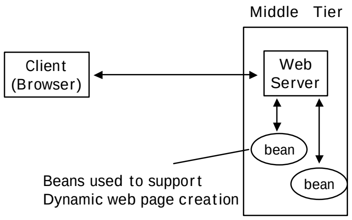
Dynamic HTML



send HTML – use its display capability and AVOID applets

Dynamic web pages
-- generated on the server

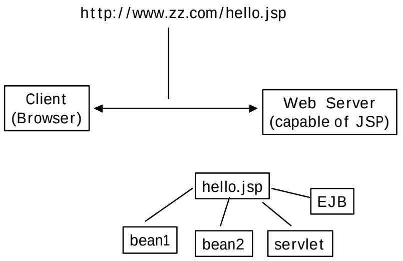
Server Side Beans



Java Server Pages (JSP)

- A kind of Web page – a combination of
  - industry-standard HTML
  - JavaServer Pages HTML tags
- The Web server takes action based on the special JSP tags
- Web server must support JSP – delivers HTML file to browser after using component services

.jsp file activation



Invoke a .jsp file – with parameters

- For example,
  - http://schnauzer:8080/simple.jsp?name=Smith

Connecting to a Bean <%= . . . %>

- Inside the JavaServer Pages file:

```
<p> The name of the row is
  <%=foobar.getRowName()%>
</p>
```

foobar is the name of the Java Bean

```
<html>
<body>
<USEBEAN  NAME=bar
   TYPE=jsp.beans.LunchSpecial  LIFESPAN=page>
<SETONCREATE BEANPROPERTY=“soup”
   VALUE=“clam chowder”>
<SETONCREATE BEANPROPERTY=“dessert”
   VALUE=“lemon meringue pie”>
</USEBEAN>
<h1>Welcome to Jake’s Emporium! </h1>
. . .
```

```
<h2>Today’s Lunch Special is: </h2>
<p>
<ul>
<li> Hot and tasty
   <DISPLAY  PROPERTY=“bar:soup”
   PLACEHOLDER=“tomato”> soup </li>
<li> Hearty
   <DISPLAY  PROPERTY=“bar:sandwich”
   PLACEHOLDER=“cheese”> sandwich </li>
<li> Homemade
   <DISPLAY  PROPERTY=“bar:dessert”
   PLACEHOLDER=“apple”> </li>
</ul>
</p>
</body>
</html>
```

Output

Welcome to Jake’s Emporium!

Today’s Lunch Special is:

```
Hot and tasty  clam chowder soup
Hearty cheese sandwich
Homemade lemon meringue pie
```

Why JavaServer Pages

- Component development is cleanly separated from Web design
- Write dynamic Web page simply
- Run on any Web server
- Access them from any Web browser
- Any server that can run Java can run JavaBeans and/or Java Servlets

Java Servlets

An alternative to CGI

Java Servlets vs. CGI

- Java Servlets written entirely in Java:
  - Write-once Run-anywhere, safe network delivery, and scalability
- Servlets execute 10 to 15 times faster than CGI on a Java Web server
- Servlets architectured to eliminate the expensive resource and performance hits of CGI

Servlet Execution

http://yourhost.com/cgi-bin-dir/servlet.sh/servlet-name

For the Servlet CGI execution engine to find your servlet, there must be a line in the servlet.properties file of the form:

- servlet.class.servlet-name=servlet-class-name

Java Server Pages or Servlets?

Design Decision

Model 1: Request to a JSP file

- Client web browser makes direct request of a JSP file
- JSP file requests information from a JavaBean
- JavaBean can in turn request information from an EJB or a database
- JavaBean generates content (perhaps working with an EJB, a database, or both)
- JSP file can query and displays the Bean's content

Model 2: Request to Java Servlet

- Client requests are handled by a Java Servlet
- The servlet generates the dynamic content
  - uses JDBC to communicate with a database to obtain the content
- The servlet wraps the dynamic content into a bean
- The JSP file accesses the dynamic content from the bean and displays the content in the client web browser

XML
Extensible Markup Language

XML

- SGML – Standard Generalized Markup Language
  – HTML – Hypertext Markup Language
  <H1>Top Heading</H1>

  – XML
  <MyTag>Top Heading<MyTag>
  + DTD – Document Type Definition

Natural Synergy

- Java
  – portable code
  – runs on any platform

- XML
  – portable data description
  – runs on any platform

DTD – Defines a Data Type

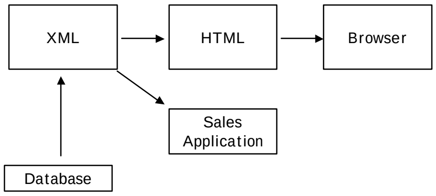
```
<!ELEMENT BILLING_PARTY
 (ACCT_Number, NAME?, ADDR?, CITY?, STATE?, ZIP?) >
```

- Name, address, city, state and zip are optional.

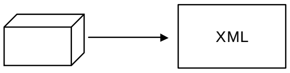
XML

```
<INVOICE>
<BILLING_PARTY>
  <ACCT_Number>Z1024</ACCT_Number>
  <NAME>John Smith</NAME>
  <ADDR>123 Elm St.</ADDR>
  <CITY>New York</CITY>
  <ZIP>10023</ZIP>
</BILLING_PARTY>
</INVOICE>
```

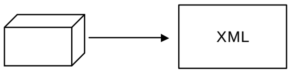
Uses for XML*



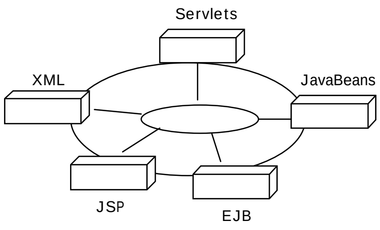
Use for XML*



XML*



Distributed Object View



END