Sun Javadays '99

EJB and OR Mapping

Dennis Leung

The Object People

THE OBJECT PEOPLE

www.objectpeople.com

About Me

Dennis Leung
Director, Product Development
The Object People
Ottawa, Canada

dennis@objectpeople.com
(613) 225-8812

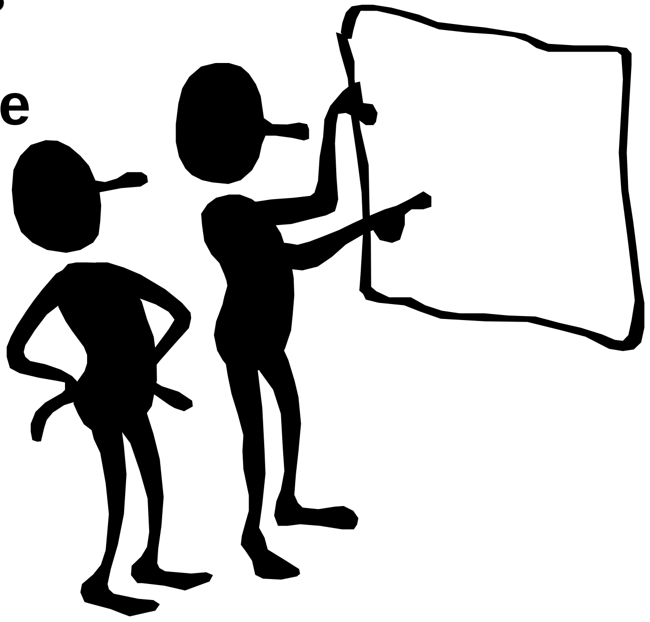
Prerequisites

- **Assumes intermediate level audience**
- **Some knowledge of EJB**
- **Basic knowledge of object-relational mapping concepts.**
    - Attributes map to columns, references to other objects may be foreign keys on the relational database.
- **Relational database and object modeling knowledge**

THE OBJECT PEOPLE

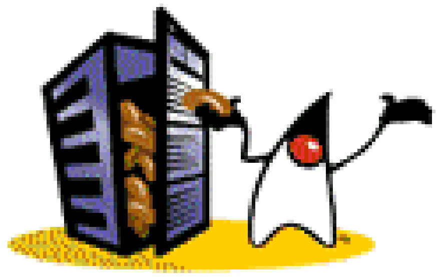
What We'll Cover

- Overview of EJB with persistence
- Session bean persistence
- Entity bean persistence
- Focus on entity-bean CMP issues
- Mostly assumes basic persistence



What is EJB?

- **Enterprise JavaBeans - a “Java Enterprise API” from Sun and its partners (IBM, Oracle, BEA…)**
- **Allows for building business logic “components” that are**
  - Distributed
  - Transactional
  - Secure



- **Often compared with CORBA and with Microsoft’s COM component architectures.**

“the standard component architecture for building distributed object-oriented business applications...”

THE OBJECT PEOPLE

What is EJB...

- **No relation to “JavaBeans”**
  - ➢ JavaBeans are client-side components.
  - ➢ Enterprise JavaBeans are server-side components.
  - ➢ Any similarity ends there.

- **Rely heavily on tools that generate the difficult code.**
  - ➢ RMI or CORBA distribution code
  - ➢ Security code based on Access Control Lists
  - ➢ Transaction code on a per-method or per class basis

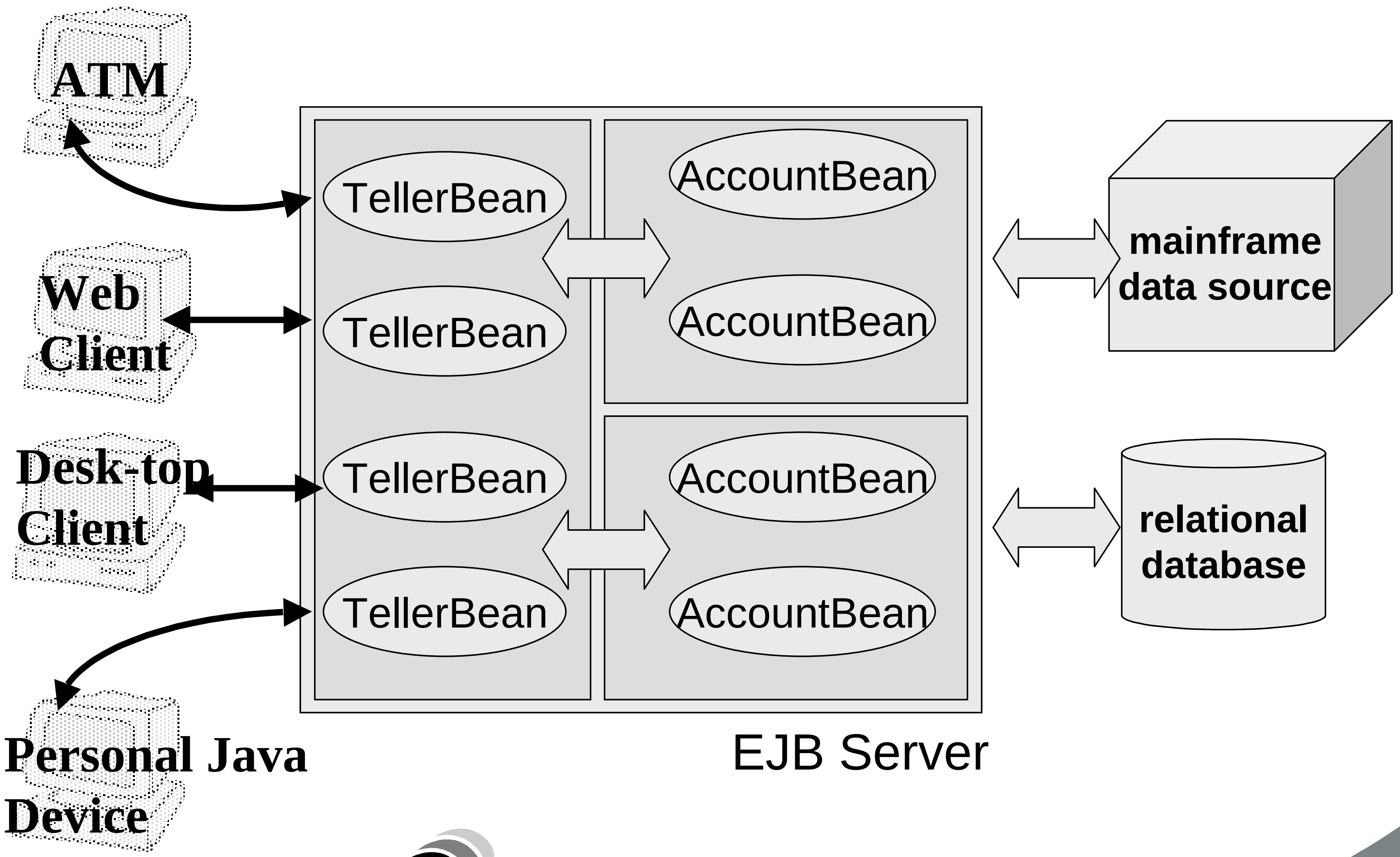
THE OBJECT PEOPLE

What is an EJB?

- **Really just domain objects that implement certain interfaces.**
  - ➢ Also have additional classes and interfaces associated with them.
- **These EJB "components" are a collection of Java classes and interfaces.**
  - ➢ A "bean" class that implements the business logic.
  - ➢ A "remote interface" that defines the client view of the bean instance.
  - ➢ A "home interface" that provides a "factory view" for creating and finding beans.
  - ➢ Additional classes may be required for some EJB servers.

THE OBJECT PEOPLE

What is an EJB?



THE OBJECT PEOPLE

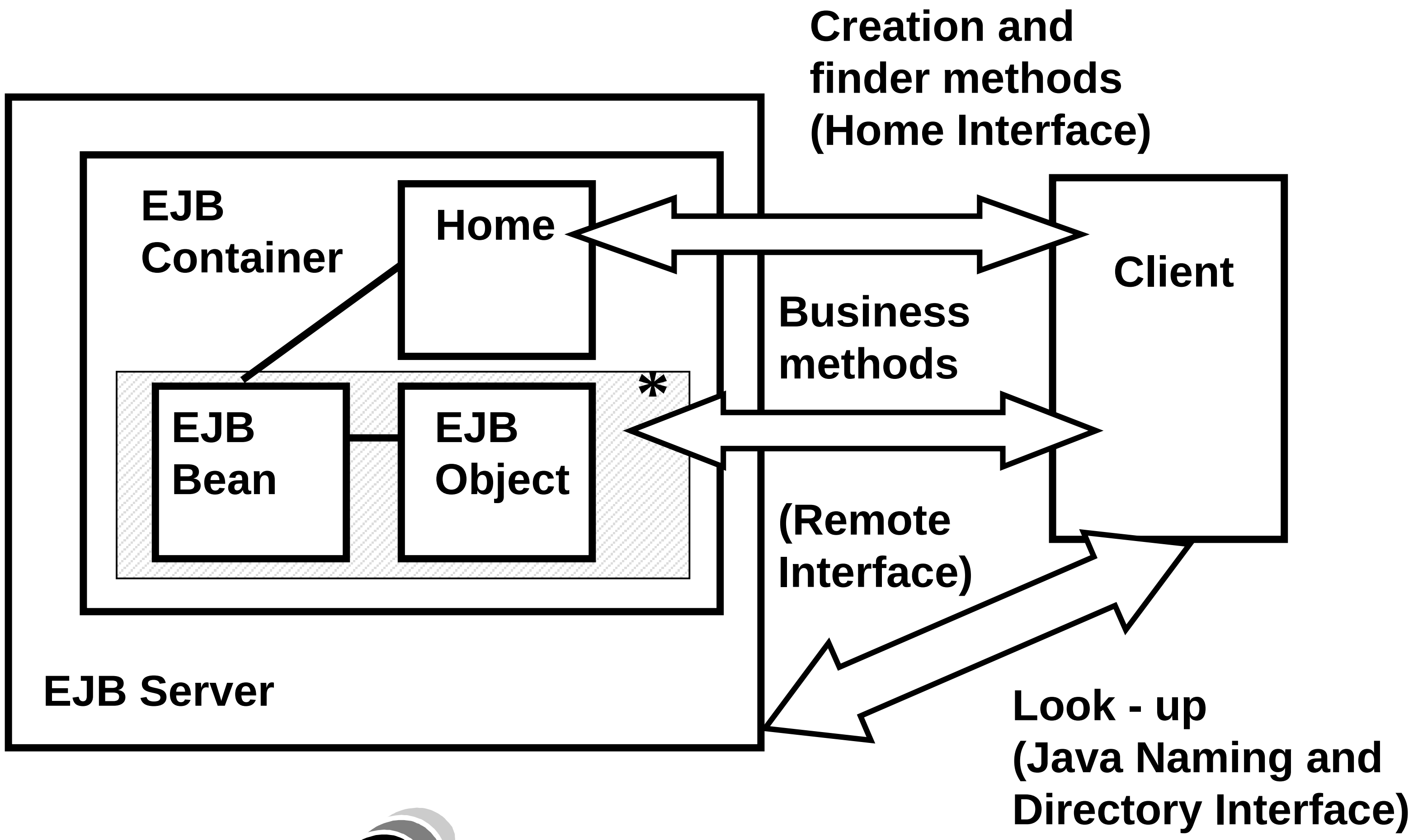
EJB Container and Server

- **The EJB Server is a “host” for the beans**
- **The Server provides services for the beans to use**
  - JNDI, JDBC connections, JMS, etc.
- **The Container provides an interface between the Server and the beans**
  - manages bean life-cycle, handles pooling or caching of beans
- **The EJB specification does not clearly define the boundary or API between the server and container**

THE OBJECT PEOPLE

EJB Architecture...

- The bean lives in a “container” in the EJB server...


Creation and finder methods (Home Interface)
EJB Container
Home
Client
Business methods
EJB Bean
EJB Object
*
(Remote Interface)
EJB Server
Look - up (Java Naming and Directory Interface)

Session and Entity Beans

- **There are two kinds of Enterprise Beans**
- **Session Beans (required in EJB 1.0)**
  - ➢ define a task, service, procedure, operation, transaction…
- **Entity Beans (optional in EJB 1.0, required in 1.1)**
  - ➢ define a persistent piece of data that resides in a relational database or some other persistent storage

Session Beans are used to implement a “business task,” while Entity Beans represent a “business entity.”

Session Beans

- **Session beans may be “stateful” or “stateless”**
  - ➢ A stateful Session bean retains information about the client that it is interacting with.
  - ➢ this state is non persistent and non-transactional
  - ➢ often referred to as “conversational state”
- **Client-specific state can be held between method calls**
- **A stateless Session bean forgets about who it is dealing with between calls.**
  - ➢ used for single requests
  - ➢ user might not get the same bean on consecutive method invocations

Session Bean Persistence

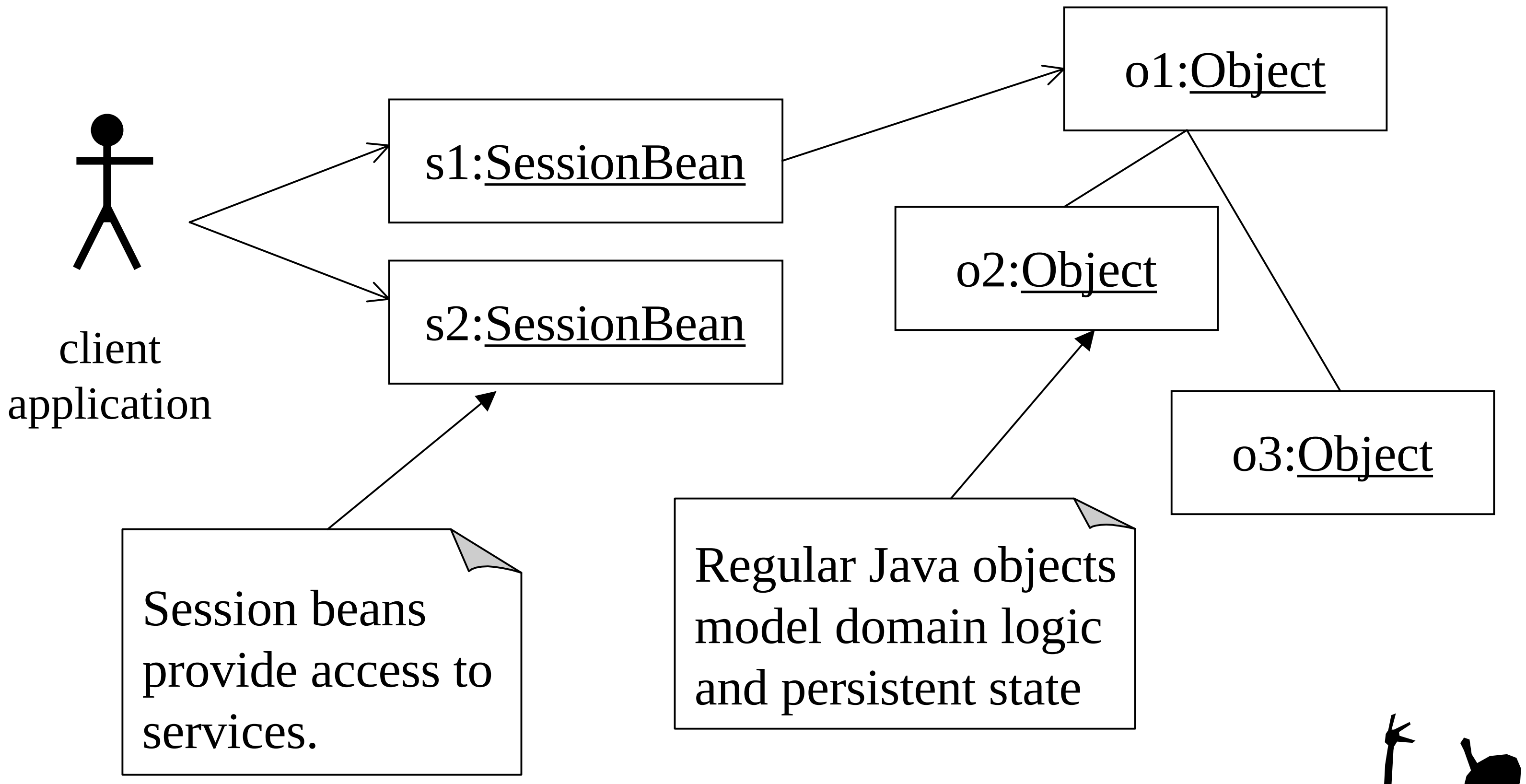
- **Session beans represent a service or operation.**
  - ➢ Do not directly represent stateful objects
  - ➢ Manipulate persitent state as entity beans, normal objects, or non-object data
- **Are often described as “coarse-grained”.**
  - ➢ May wrap a non-Java program.
- **Popular examples of Session beans are:**
  - ➢ Shopping cart
  - ➢ Banking Services (Teller)
  - ➢ Reservation System

THE OBJECT PEOPLE

Pure Session Bean Architecture...

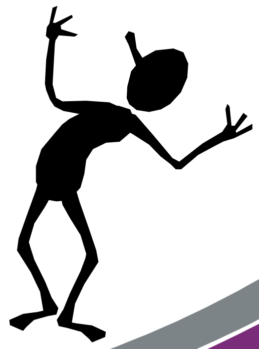
- **Session beans carry out all of the server-related operations.**
- **Persistent data modeled using regular Java objects and some persistence mechanism.**
- **Benefits:**
  - simple architecture
  - fast access times
  - little additional infrastructure needed
  - few limitations on domain model
- **Drawbacks**
  - simple client behavior
  - no “real” objects at the bean level
  - transactions must be managed for non-beans

Pure Session Bean Architecture



Session beans provide access to services.

Regular Java objects model domain logic and persistent state



Entity Beans - Persistence

- **Persistence is the central feature of Entity EJBs.**
- **How persistence is achieved is not described in the EJB spec.**
- **All EJB persistence is “automatic” as far as the user of the bean is concerned.**
    - the bean client never has to explicitly store the bean
    - timing of load and store is left to the EJB Server/Container
- **EJB persistence is generally assumed to be through relational databases, although it can take other forms**
    - Object database, file system, proprietary storage system

Entity Beans and Databases

- **Simple model**
  - ➢ Relationships between beans not discussed
  - ➢ Basic “one bean = one row in one table” mapping to database world.
  - ➢ No standard querying language.
- **To remain independent of how the beans are actually stored, EJB presents a very basic view of persistence.**

Entity Beans - BMP/CMP

- **Entity Beans are persistent domain objects - two persistence mechanisms possible:**
  - Bean-managed persistence (BMP)
  - Container-managed persistence (CMP)
- **With BMP, the developer writes their own persistence code directly in the bean.**
- **With CMP persistence is declarative, based on information provided at deployment time.**
  - this is often referred to as “automatic” persistence

THE OBJECT PEOPLE

Entity Beans - BMP

- **Bean-managed persistence "lets" developers write the persistence code themselves.**
  - ➢ Dictates how persistence is to be handled.
- **Database reads and writes occur in specific methods defined for bean instances.**

```
ejbLoad() - "load yourself"
ejbStore() - "store yourself"
ejbCreate() - "create yourself"
findBy…() - "find yourself"
ejbRemove() - "remove yourself"
```

- **The Server or Container decides when these methods are called.**

THE OBJECT PEOPLE

Entity Beans - BMP (continued)

- **BMP allows code to be custom-written for specific sitautions: can hand-tune, target less common platforms**
- **Having persistence code directly in the bean instance leads to some problems.**
  - Object identity not guaranteed - findOne is an instance method, can defeat cache hits
  - Efficiency - findMany returns only primary keys, so each bean requires a separate database read
  - Limited control - user has control over direct persistence, but not related issues (caching, locking, concurrent access)
  - “Manual” relationship management

THE OBJECT PEOPLE

Entity Beans - CMP

- **With container-managed persistence persistence is based on information in the deployment descriptor.**
  - ➢ Different kinds of persistence mechanisms will require different "containers" that will provide the right code for the beans.
- **Persistence is "automatic" not only for the user of the bean, but also for the developer.**
  - ➢ No persistence code needs to be written in the bean in this case.
  - ➢ Code may be generated in the container or persistence may be based on meta-data.

THE OBJECT PEOPLE

Entity Beans - CMP (continued)

- **Existing persistence frameworks cannot immediately be used for container-managed persistence.**
  - ➢ Requires integration with EJB Server.
  - ➢ No standard API has been defined for EJB Server interactions, therefore each integration is specialized.
- **Issues**
  - ➢ Cache integration
  - ➢ Bean Relationships
  - ➢ Queries
  - ➢ Bean Inheritance
  - ➢ Transactions
  - ➢ Concurrent access
  - ➢ Database integrity constraints
  - ➢ One table/one class assumption

THE OBJECT PEOPLE

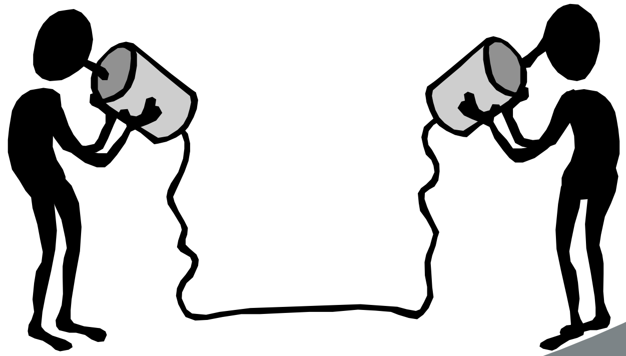
Cache Integration

- **EJB Server provides a cache**
- **Persistence frameworks provide a cache**
- **Caches must be kept in sync or combined**
  - ➢ Multiple copies vs. in-memory locking
  - ➢ Avoids caching problems of BMP
    - cache hits can occur even on non-PK queries
    - reads of multiple beans can instantiate and return beans directly

THE OBJECT PEOPLE

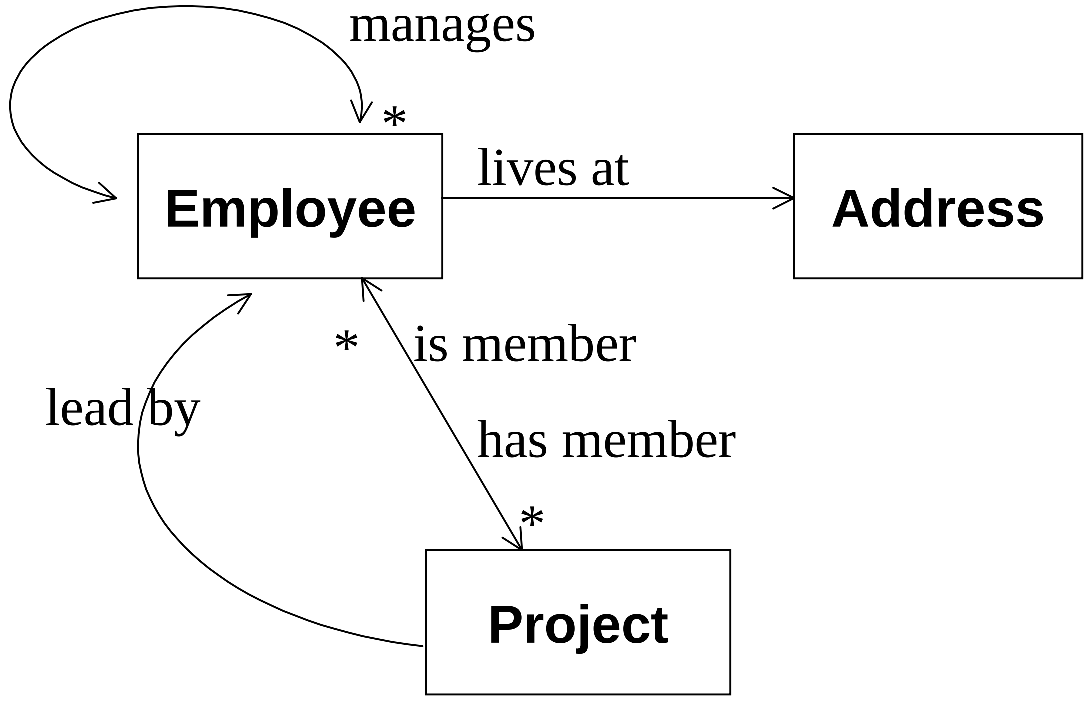
Bean Relationships

- **The EJB specification does not discuss how Entity beans should be related to one another.**
  - ➢ EJB 2.0 is planned to address relationships
  - ➢ We assume the simplest mechanism for relationships, through remote interfaces
- **Relationships exist between "remote interfaces."**
  - ➢ No special relationship objects
  - ➢ No persistent relationship management code in bean.
  - ➢ Same mechanism as relationships from clients to beans



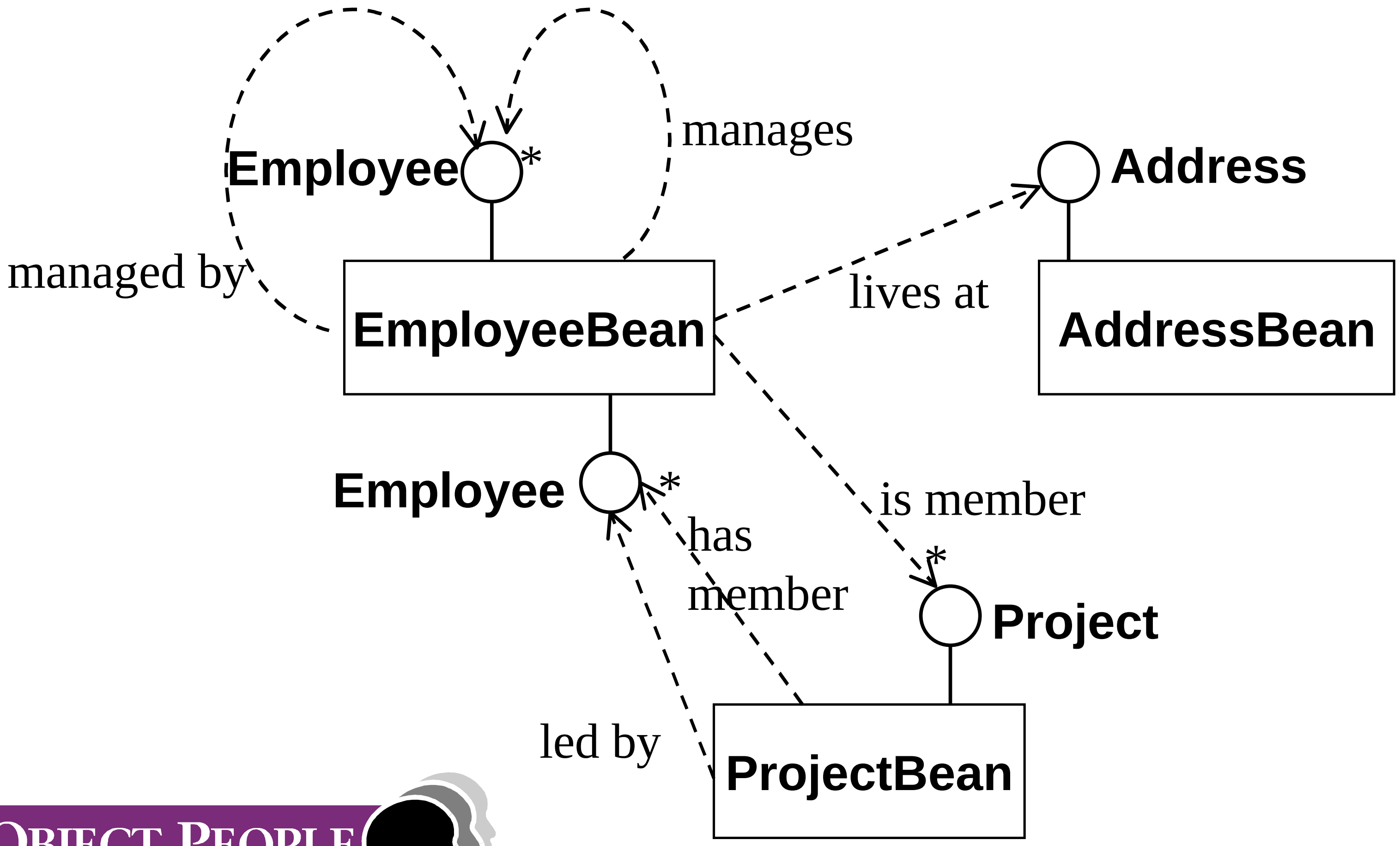
Bean Relationships...

- **In a normal domain model, domain objects refer to each other directly.**



Bean Relationships...

- **In EJB, domain objects (Entity beans) must refer to one another using their remote interfaces.**



THE OBJECT PEOPLE

Bean Relationships...

- **If EmployeeBean is to be related to an AddressBean, it must refer to the bean through its remote interface.**

```
public class EmployeeBean implements EntityBean {
            public EntityContext ctx; // required by EJB 1.0
            public Address address; // remote interface for AddressBean
            //...
}
```

Finders and Queries

- **Reading objects is defined in terms of “finder” methods on the home.**
- **No standard (portable) way of defining a finder method**
  - Hand-written code based on natural language description
  - Proprietary finder description language (not portable between servers/containers)
  - Specify directly in terms of underlying database (e.g. SQL)
- **Finders represent static queries**
  - No ad hoc/dynamic querying
- **Beans are heavy-weight components**
  - Sometimes you just want data

Finders and Queries (continued)

- **Want an extensible querying system**
  - In terms of objects, not rows
  - expressive (joins, inheritance, multiple tables, …)
  - static or dynamic (findByQuery)
  - support querying for raw data as well as beans
  - anything the container can do, I can do…
- **Non-proprietary would be nice but…**
  - OQL: problematic for relational DB, not widely used
  - SQL3: poor match for object model
  - non-string format?: bad for non-Java clients
  - others?

THE OBJECT PEOPLE

Bean-Level Queries

- **Queries should be specified in terms of the object model, not in terms of rows**
  - employee.manager.address = someAddress
  - SELECT * FROM EMP t1, EMP t2, ADDR t3 WHERE t1.MGR_ID = t2.EMP_ID AND t2.ADDR_ID = t3.ADDR_ID AND t3.ADDR_ID = <someAddress.id>
- **This is non-trivial**
  - joins, self-joins
  - multiple tables/multiple objects in a row
  - inheritance
  - database functions (employee.name.toUppercase())

THE OBJECT PEOPLE

Data-Level Queries

- **Instantiating beans is expensive**
  - heavyweight objects, remote interface, caching
- **You don't always need the full objects**
  - displaying a list
  - performing simple calculations
- **Support data-level reads**
  - specific fields
  - aggregate functions
  - unmapped data

THE OBJECT PEOPLE

Inheritance

- **How do we represent inheritance in a relational database?**
  - ➢ each abstract and sub-class has its own table
  - ➢ each sub-class has data in its own table
  - ➢ sub-classes have data in its own table as well as parent’s
- **Breaks the 1class/1 table idea**
- **Need a type column (or something…) to distinguish sub-class.**
- **Super-classes need to be able to find all sub-classes.**

THE OBJECT PEOPLE

EJB and Inheritance

- **Inheritance is not mentioned in EJB 1.0.**
  - ➢ Is mentioned in EJB 1.1 but not dealt with…
  - ➢ Varies by server
- **Typically inheritance can be used as follows:**
  - ➢ homes do not inherit
  - ➢ beans can inherit from one another
  - ➢ remote interfaces can inherit from one another
- **The notion of “component inheritance” is not clearly defined.**

THE OBJECT PEOPLE

EJB and Transactions

- **EJB has “declarative” transactions**
  - ➢ normally delimited by start/end of method calls
- **Ideally, want full transactional semantics at the bean level**
  - ➢ What objects participate?
    - The server knows, beans register as synchronized
  - ➢ What needs to be written?
    - Need to track changes
  - ➢ “Transaction” should control object writes
    - Re-order writes to respect integrity constraints
  - ➢ Rollback or discard beans on commit failure

THE OBJECT PEOPLE

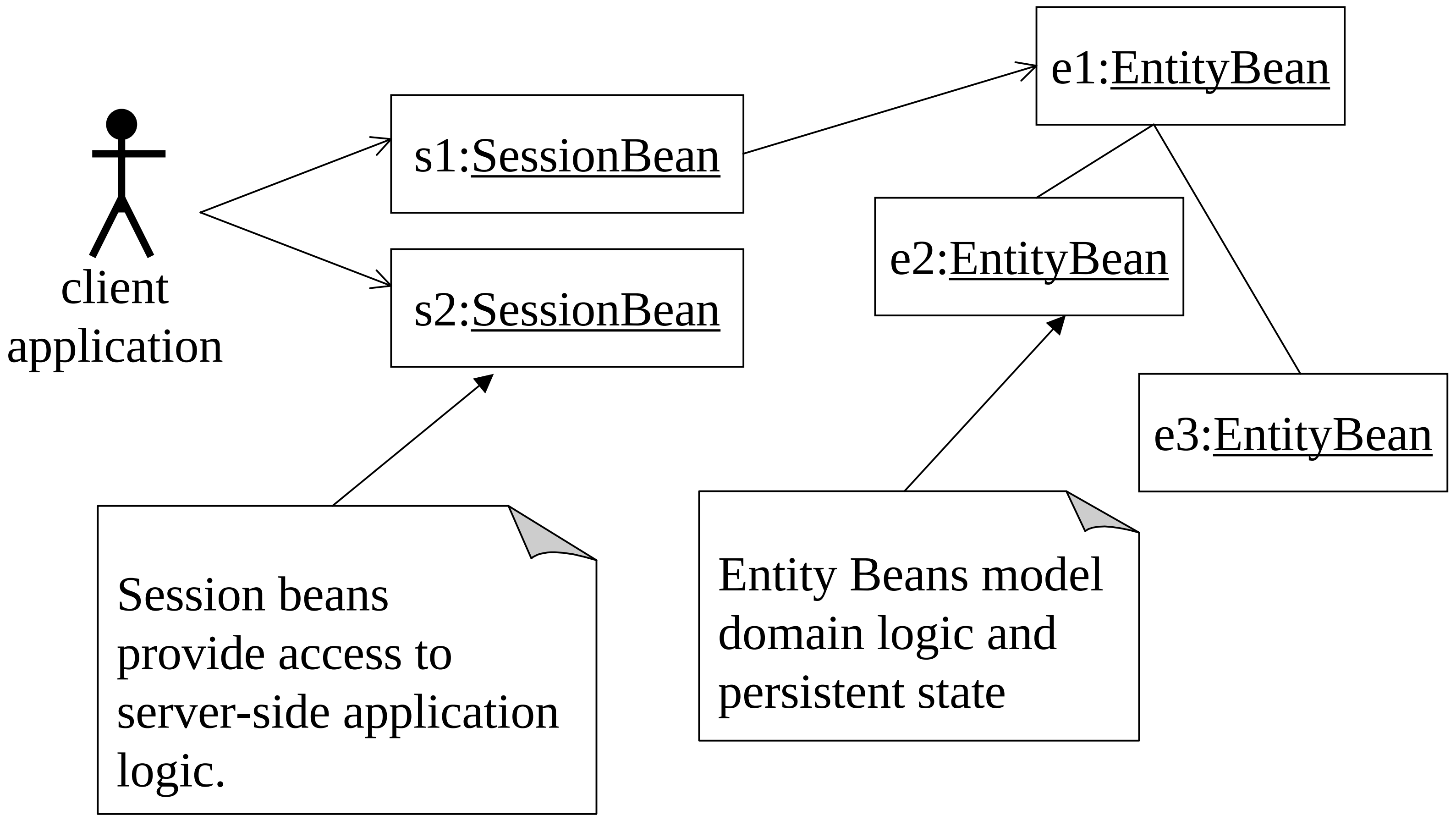
Entities and Transactions…

- **Exercise care with container-managed transactions.**
  - Default behaviour is a separate transaction for every method call.
  - Reasonable for session bean “services”, not normally reasonable for entity beans
  - Transactions are expensive, transaction semantics are usually
  - Leaving transaction management up to the client is not necessarily the best idea.
- **So...**

Session & Entity Tiered Architecture

- **Client access is limited to Session beans, which in turn access Entities.**
- **Entity beans are used to model persistent domain entities.**
- **Benefits:**
  - ➢ Session beans provide transactions and security while Entity beans provide persistence mechanism.
  - ➢ Uses the strengths of both types of beans.
- **Drawbacks:**
  - ➢ Greater complexity/overhead on domain model.

Session & Entity Tiered Architecture…



Session beans provide access to server-side application logic.

Entity Beans model domain logic and persistent state

Concurrent Access

- **Transaction cannot be allowed to interfere**
- **Pessimistic locking of beans too restrictive**
- **Make copies**
  - ➢ Each transaction has a separate copy of beans
  - ➢ Manage access at the database level
    - Pessimistic locking: for complete certainty
      - – Usually not appropriate for interactive applications
    - Optimistic locking: better performance, concurrency

Application Server Performance

- **Server Optimizations**
  - ➢ Resource sharing/pooling
    - JDBC connection pooling
    - Shared cache for read-only objects
  - ➢ Replication
    - EJBs are pure server-side objects
    - Migration to client can be a big win
    - Session bean architecture?
    - Migrate copies of entity bean data, push back?

Application Server Performance

- **Bean Granularity**
  - ➢ All inter-bean calls go through remote invocation
  - ➢ Non-reentrant, restrictive on domain model
  - ➢ Consider coarser-grained entity beans with “dependent objects” (EJB 1.1 terminology)
    - Persistence framework must handle mixed beans/non-beans
    - Dependent objects can only be passed by value

THE OBJECT PEOPLE

Persistence Optimizations

- **Optimized Reading**
  - ➢ Minimize database round-trips
    - Read multiple objects at a time (findMany, joins)
    - Data-level reads
    - Do work in the database
  - ➢ Avoid reading too much
    - Database cursors
  - ➢ Stored procedures/Static SQL
  - ➢ Clever use of caching

THE OBJECT PEOPLE

Summary

- **A number of architectures for EJB, each have their own set of issues related to object/relational persistence.**
  - Session Beans with persistent Java objects
  - Session Beans with Entity beans
  - Bean-managed Entity beans
  - Container-managed Entity beans