© Copyright 2004, Oracle Corporation

Evolution of the EJB Entity

Michael Keith

Oracle Corporation

mkeith@oracle.com

© Copyright 2004, Oracle Corporation

About Me

- 14+ years experience in OO persistence
- Worked on many projects, including one of the earliest EJB (1.0) CMP implementations
- Technical Architect for OracleAS TopLink and OracleAS EJB Container in OC4J
- Currently on JSR 220 (EJB 3.0) expert group

© Copyright 2004, Oracle Corporation

Audience Poll

1. How many people have used EJB entities?
2. How many have used them since EJB 1.1 or earlier?
3. How many were/are happy with them?
4. How many people use other persistence solutions and are happy with those?

© Copyright 2004, Oracle Corporation

Agenda

- Setting the Stage
- A New Component Model
- The Entity Requirement
- Updating the Model
- Filling in Holes
- The Golden Age of Simplification
- Summary

© Copyright 2004, Oracle Corporation

Agenda

- Setting the Stage
- A New Component Model
- The Entity Requirement
- Updating the Model
- Filling in Holes
- The Golden Age of Simplification
- Summary

© Copyright 2004, Oracle Corporation

Life Before EJB

In a galaxy far far away…

- Emergence of middle-tier Java architectures
- Lack of standard mechanisms for encapsulating and specifying business logic
- No organized cohesive technology base for enterprise Java (pre-J2EE)
- Developers had to keep re-inventing the server-side wheel of transactions, concurrency, and security
- Community acquired the vision of pluggable server-side Java components

© Copyright 2004, Oracle Corporation

Birth of the Bean

- April 1997 – Sun announces Enterprise JavaBean technology initiative
- Specification group created from people at Sun as well as input from its partner companies
- Ongoing specification effort for a year
- During this time EJB was heralded by some as the best new technology to ever hit the server side (without even trying it out)
- March 1998 Enterprise JavaBeans 1.0 released

© Copyright 2004, Oracle Corporation

Goals of EJB

The initial goals of EJB were:

- Allow components developed separately to be deployed together and interoperate in the server
- Define development and deployment contracts so that the development tools can produce interoperable components
- Lessen the knowledge required to develop components
- Provide access to low-level APIs for advanced developers
- *"Write once, run on any EJB Container!"*
- Interoperability with non-Java applications; compatibility with CORBA

© Copyright 2004, Oracle Corporation

EJB Timeline

Date	Event
Feb 1997	JDK 1.1
Apr 1997	EJB announced
Mar 1998	EJB 1.0
Dec 1998	Java 2 (JCP started)
Jun 1999	J2SE, J2EE announced
Dec 1999	EJB 1.1 (J2EE 1.2)
May 2000	J2SE 1.3
Sep 2001	EJB 2.0 (J2EE 1.3)
Feb 2002	J2SE 1.4
Apr 2004	EJB 2.1 (J2EE 1.4)
Sep 2004	J2SE 5.0

© Copyright 2004, Oracle Corporation

Agenda

- Setting the Stage
- A New Component Model
- The Entity Requirement
- Updating the Model
- Filling in Holes
- The Golden Age of Simplification
- Summary

© Copyright 2004, Oracle Corporation

EJB 1.0

- "Roles" of those involved from the development of the bean to its usage in an application
- Contracts between the Bean Provider, the EJB Container and the client
- Life cycle model
- Declarative transaction demarcation, JTS
- Distribution, location transparency
- Security using role-based identities
- Environment properties available to bean at runtime
- Descriptor class API (serialized deployment descriptor)
- Entity beans optional for EJB Container implementors

© Copyright 2004, Oracle Corporation

Roles

Six different roles in the process from bean development to bean deployment within an EJB system:

1. Bean Provider
2. Application Assembler
3. Deployer
4. Server Provider
5. Container Provider
6. System Administrator

Some of these may be fulfilled by the same person

© Copyright 2004, Oracle Corporation

Roles

I am a domain expert

So am I, except I have to know more than just EJB

I have to know all about the deployment platform, cuz' I have to deploy the app

Bean Provider

Application Assembler

Deployer

Tools

I monitor the performance and integrity of the app at runtime

Container Provider

Server Provider

System Administrator

© Copyright 2004, Oracle Corporation

Component Contract

Contract between an entity and the Container it runs in

Comprises:

- Life cycle and state callbacks (on javax.ejb.EntityBean interface) occur at the correct time
- Container provides the EntityContext object and implements the methods correctly on it
- Container provides the other lower level services (normally part of the server) such as the transactions, data sources, security, *etc*.

© Copyright 2004, Oracle Corporation

Client View Contract

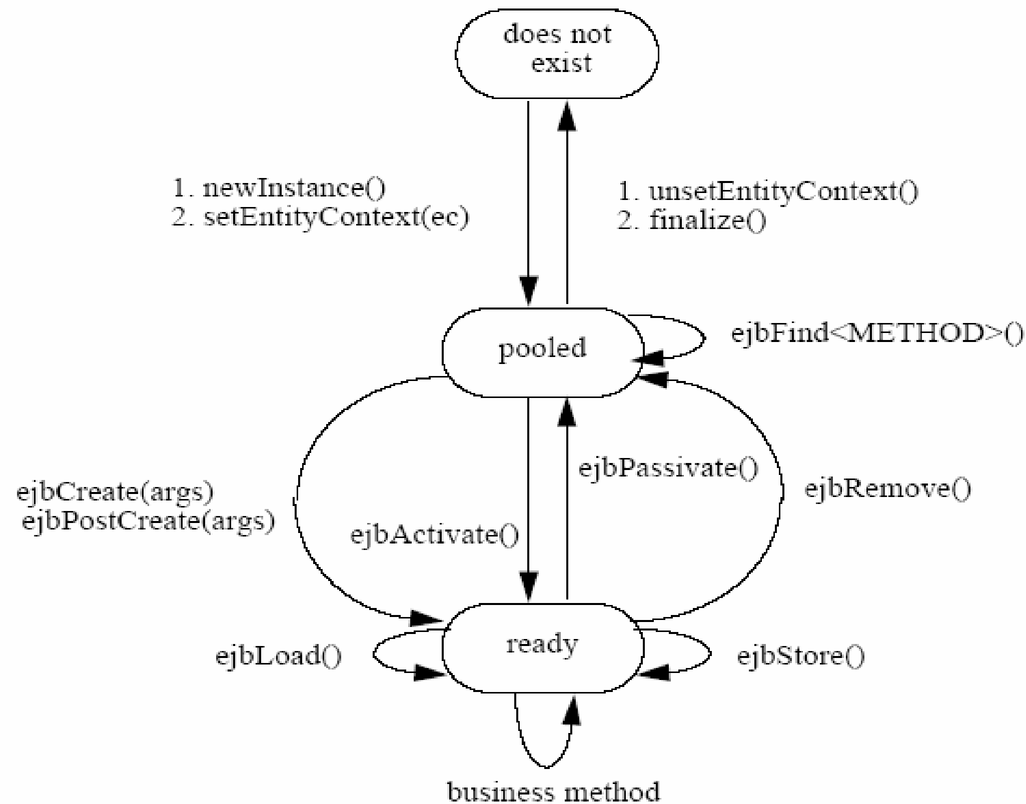
Contract between a client and the Container/Bean Provider

Comprises:

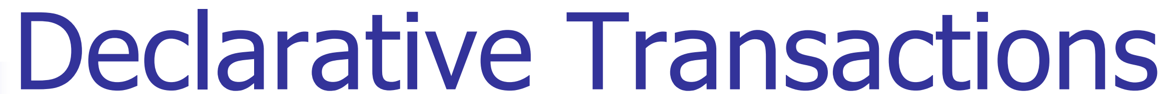
- Each entity has a unique identity (assigned at create-time)
- Container posts entity home objects in JNDI under the deployer-supplied JNDI names
- Bean provider must provide the home and remote interfaces (extending EJBHome/EJBObject)
- Container must generate implementation classes and facilitate clients being able to remotely invoke them

© Copyright 2004, Oracle Corporation

Life Cycle



© Copyright 2004, Oracle Corporation



Declarative Transactions

- Java Transaction Service (JTS) – Java binding of CORBA Object Transaction Service (OTS) part of the contract
- JDBC transaction isolation options
- Transaction attributes:
  - TX_NOT_SUPPORTED
  - TX_SUPPORTS
  - TX_BEAN_MANAGED
  - TX_REQUIRES_NEW
  - TX_REQUIRED
  - TX_MANDATORY
- Use `javax.jts.UserTransaction` when TX_BEAN_MANAGED

© Copyright 2004, Oracle Corporation

Distribution

- All references are remote (RMI objects)
- Lookups of homes return EJBHome RMI stubs and operations on the homes return EJBObject RMI stubs
- Remote stubs and skeletons are generated by EJB compiler
- Interoperability achieved by generating CORBA IIOP stubs
- Standard approach is to “narrow” a stub using `javax.rmi.PortableRemoteObject.narrow()`
- Optional other protocol support by Containers

© Copyright 2004, Oracle Corporation

Security

- User/role-based security integrates with `java.security.Identity` in SDK
- Platform-specific way to map Identity to users/roles
- Manual security checks through EJBContext calls like `getCallerIdentity()` and `isCallerInRole()`
- `AccessControlEntry` in deployment descriptor to declaratively state which identities are allowed to invoke the bean methods
- For entities, all methods in the same transaction should have matching roles

© Copyright 2004, Oracle Corporation

Environment Properties

- Set of key/value pairs that the Bean Provider/Deployer provides with the ejb-jar file
- Regular `java.util.Properties` object stored as part of the deployment descriptor metadata
- Bean instances access the environment properties through the `EntityContext.getEnvironment()` method

© Copyright 2004, Oracle Corporation

Descriptors

- Set of specification-defined classes that standardize the bean metadata
- Deployment descriptor specified as being simply the serialized form of these classes
- Not text-editable for modification during deployment

`EntityDescriptor` – extends more general DeploymentDescriptor class

`ControlDescriptor` – transaction metadata and runAs security settings

`AccessControlEntry` – method-level security metadata

© Copyright 2004, Oracle Corporation

```
public class javax.ejb.deployment.DeploymentDescriptor
     extends java.lang.Object
     implements java.io.Serializable {

public ControlDescriptor[] getControlDescriptors();
public void setControlDescriptors(ControlDescriptor value[]);

public AccessControlEntry[] getAccessControlEntries();
public void setAccessControlEntries(AccessControlEntry values[]);

public Properties getEnvironmentProperties();
public void setEnvironmentProperties(Properties value);
```

© Copyright 2004, Oracle Corporation

DeploymentDescriptor

```
public boolean isReentrant();
public void setReentrant(boolean value);

public Name getBeanHomeName();
public void setBeanHomeName(Name value);

public String getEnterpriseBeanClassName();
public void setEnterpriseBeanClassName(String value);

public String getHomeInterfaceClassName();
public void setHomeInterfaceClassName(String value);

public String getRemoteInterfaceClassName();
public void setRemoteInterfaceClassName(String value);
}
```

© Copyright 2004, Oracle Corporation

EntityDescriptor

```
public class javax.ejb.deployment.EntityDescriptor
     extends javax.ejb.deployment.DeploymentDescriptor {

public Field[] getContainerManagedFields();
public void setContainerManagedFields(Field values[]);

public String getPrimaryKeyClassName();
public void setPrimaryKeyClassName(String value);
}
```

ControlDescriptor

```
public class javax.ejb.deployment.ControlDescriptor
      extends java.lang.Object
      implements java.io.Serializable {

public final static int CLIENT_IDENTITY;
public final static int SPECIFIED_IDENTITY;
public final static int SYSTEM_IDENTITY;

public final static int TRANSACTION_READ_COMMITTED;
public final static int TRANSACTION_READ_UNCOMMITTED;
public final static int TRANSACTION_REPEATABLE_READ;
public final static int TRANSACTION_SERIALIZABLE;

public final static int TX_BEAN_MANAGED;
public final static int TX_MANDATORY;
public final static int TX_NOT_SUPPORTED;
public final static int TX_REQUIRED;
public final static int TX_REQUIRES_NEW;
public final static int TX_SUPPORTS;
```

© Copyright 2004, Oracle Corporation

ControlDescriptor

```
public Method getMethod();
public void setMethod(Method value);

public int getRunAsMode();
public void setRunAsMode(int value);
public Identity getRunAsIdentity();
public void setRunAsIdentity(Identity value);

public int getTransactionAttribute();
public void setTransactionAttribute(int value);

public int getIsolationLevel();
public void setIsolationLevel(int value);
}
```

© Copyright 2004, Oracle Corporation

AccessControlEntry

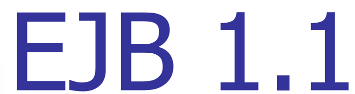
```
public class javax.ejb.deployment.AccessControlEntry
     extends java.lang.Object
     implements java.io.Serializable

public Identity[] getAllowedIdentities();
public void setAllowedIdentities(Identity values[]);

public Method getMethod();
public void setMethod(Method value);
}
```

© Copyright 2004, Oracle Corporation

Agenda

- Setting the Stage
- A New Component Model
- The Entity Requirement
- Updating the Model
- Filling in Holes
- The Golden Age of Simplification
- Summary

© Copyright 2004, Oracle Corporation



EJB 1.1

- Entity support mandatory for EJB Containers
- XML deployment descriptor introduced – API classes deprecated
- Better separation of Bean Provider/Application Assembly entries in deployment descriptor
- Security – java.security.Principal (with Java 2)
- Environment – specified in deployment descriptor and accessible at runtime through JNDI
- Transaction – specification generalized to JTA
- Container-generated primary key

© Copyright 2004, Oracle Corporation

Deployment Descriptor

- XML format with DTD for validation
- Divided into two major parts to better define the Bean Provider and Application Assembly roles
- Structural information – Bean Provider
  - Basic metadata relating to interfaces/classes, external dependencies, transaction/persistence type
- Assembly information – Application Assembler
  - Metadata relating to security roles, method permissions, transaction attributes

© Copyright 2004, Oracle Corporation

Deployment Descriptor

```
<ejb-jar>
    <enterprise-beans>
        <entity>
            <ejb-name>Order</ejb-name>
            <home>com.acme.OrderHome</home>
            <remote>com.acme.Order</remote>
            <ejb-class>com.acme.OrderBean</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>java.lang.Integer</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-field><field-name>orderId</field-name></cmp-field>
            <cmp-field><field-name>state</field-name></cmp-field>
            <primkey-field>orderId</primkey-field>
        </entity>
    </enterprise-beans>
```

© Copyright 2004, Oracle Corporation

Deployment Descriptor

```
    <assembly-descriptor>
        <security-role><role-name>admin</role-name></security-role>
        <method-permission>
            <role-name>admin</role-name>
            <method>
                <ejb-name>Order</ejb-name>
                <method-name>changeOrderState</method-name>
            </method>
        </method-permission>
        <container-transaction>
            <method>
                <ejb-name>Order</ejb-name>
                <method-name>*</method-name>
            </method>
            <trans-attribute>Required</trans-attribute>
        </container-transaction>
    </assembly-descriptor>
</ejb-jar>
```

© Copyright 2004, Oracle Corporation

Environment

- Define environment entries in structural section of the deployment descriptor
- Bean can determine values at runtime by looking up the entries in naming sub-context `java:comp/env` of JNDI
- Three different types of environment entries:

1. Simple environment variables with supplied values
2. References to other beans
3. References to resource factories

© Copyright 2004, Oracle Corporation

Environment Variables

Environment variable declared in descriptor:

```
<env-entry>
    <env-entry-name>executionMode</env-entry-name>
    <env-entry-type>String</env-entry-type>
    <env-entry-value>Production</env-entry-value>
</env-entry>
```

Variable value can be looked up in bean code:

```
Context namingCtx = new InitialContext();
String mode = (String)initCtx.lookup("java:comp/env/executionMode");
if (“Development”.equals(mode)) {
    System.out.println(“Running in development mode”);
}
```

© Copyright 2004, Oracle Corporation

Bean References

Bean reference declared in descriptor:

```
<ejb-ref>
    <ejb-ref-name>ejb/ItemObject</ejb-ref-name>
    <ejb-ref-type>Entity</ejb-ref-type>
    <home>org.acme.ItemHome</home>
    <remote>org.acme.Item</remote>
    <ejb-link>Item</ejb-link>
</ejb-ref>
```

Bean home can be looked up in bean code:

```
Context namingCtx = new InitialContext();
Object obj = namingCtx.lookup("java:comp/env/ejb/ItemObject");
ItemHome itemHome = (ItemHome)
    PortableRemoteObject.narrow(obj, ItemHome.class);
```

© Copyright 2004, Oracle Corporation

Resource References

Resource factory reference declared in descriptor:

```
<resource-ref>
    <res-ref-name>jdbc/OrderDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>
```

Resource factory can be looked up in bean code:

```
Context namingCtx = new InitialContext();
javax.sql.DataSource ds = (DataSource)
    namingCtx.lookup("java:comp/env/jdbc/OrderDB");
java.sql.Connection conn = ds.getConnection();
```

© Copyright 2004, Oracle Corporation

Transactions

- Changed to use JTA instead of JTS, so no guarantees of transaction context propagation
- Changed transaction attributes:
  - Removed TX_BEAN_MANAGED and created a separate `persistence-type` element with valid values of `Bean` or `Container`
  - Added Never attribute to disallow transaction context on method invocation
- Removed JDBC transaction isolation settings

© Copyright 2004, Oracle Corporation

Agenda

- ➢Setting the Stage
- ➢A New Component Model
- ➢The Entity Requirement
- ➢Updating the Model
- ➢Filling in Holes
- ➢The Golden Age of Simplification
- ➢Summary

EJB 2.0

- Bean Provider writes beans as abstract classes with abstract getter/setter methods
- Standardized finder definitions
- EJB Query Language for query criteria (EJB QL)
- Internal ejbSelects methods that can return different bean types as well as simple data attributes
- Home interface methods that do not need an instance of any particular identity to execute on
- Local homes/interfaces for optimized intra-VM components
- Complete specification of relationships between entities

Colorado Software Summit

© Copyright 2004, Oracle Corporation

Abstract Bean Classes

```
import javax.ejb.*;
public abstract class OrderBean implements EntityBean {

//    public Integer orderId;
//    public Integer state;

    public OrderBean() {}

    public Integer ejbCreate(Integer orderId) {
        this.setOrderId(orderId);
        this.setState(null);
        return null;
    }
    public void ejbPostCreate(Integer orderId) {}
```

© Copyright 2004, Oracle Corporation

Abstract Bean Classes

```
public void ejbActivate() {}
public void ejbLoad() {}
public void ejbPassivate() {}
public void ejbRemove() javax.ejb.RemoveException {}
public void ejbStore() throws javax.ejb.EJBException {}
public void setEntityContext(EntityContext ctx) {}
public void unsetEntityContext() {}

public abstract Integer getOrderId();
public abstract void setOrderId(Integer orderId);
public abstract Integer getState();
public abstract void setState(Integer state);
// . . .  plus any business logic . . .
}
```

© Copyright 2004, Oracle Corporation

Local Homes/Interfaces

- Added EJBLocalHome and EJBLocalObject interfaces
- Indicates that the Container can optimize for locality (client and bean are co-located in the same VM)
- Assume bean is local, so location transparency is gone
- Beans are not RMI objects and can be passed by reference (Caution: Shared objects can bite the app)
- No CORBA remoteness, so no type narrowing required
- Less overhead, entities can represent more fine-grained data than before
- EJBLocalObject component interface methods do **not** throw java.rmi.RemoteException

© Copyright 2004, Oracle Corporation

EJB QL

- SQL-like with relational constructs but uses Objects
- Basic SELECT … FROM … WHERE structure
- Instead of tables use `abstract-schema-name` to represent object types
- Can parameterize WHERE clause using "?" notation
- Navigate to cmp-fields and cmr-fields using "." notation
- Can return beans, cmp-field or collection of cmp-fields
- Support for some, but not all of the SQL constructs
- Support for some built-in functions

© Copyright 2004, Oracle Corporation

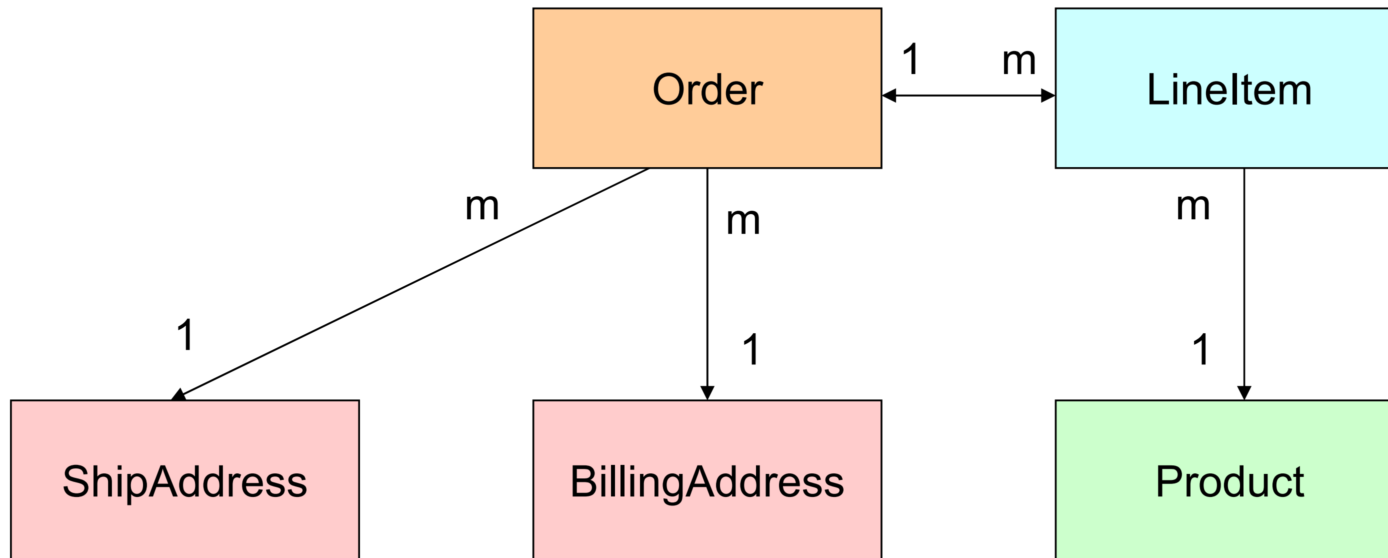
EJB QL

Reserved words/tokens in EJB QL 2.0:

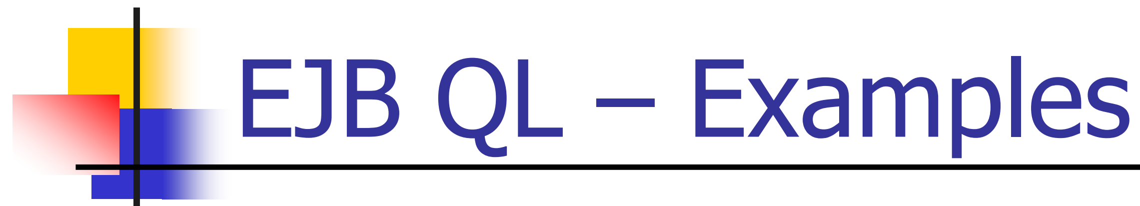
- Structural – SELECT, FROM, WHERE, OBJECT
- Range variables, collection member declarations – IN, AS
- DISTINCT
- Comparison operators – IS, EMPTY, MEMBER, OF, IN, BETWEEN, LIKE, NULL, =, >, >=, <, <=, <>
- Arithmetic operators – +, -, *, /
- Literals – TRUE, FALSE

© Copyright 2004, Oracle Corporation

A Model



© Copyright 2004, Oracle Corporation



EJB QL – Examples

Find all orders:

```
SELECT OBJECT(o) FROM Order o
```

Find all orders that need to be shipped to California:

```
SELECT OBJECT(o) FROM Order o WHERE o.shipAddress.state = ‘CA’
```

Find all states for which there are orders:

```
SELECT DISTINCT o.shipAddress.state FROM Order o
```

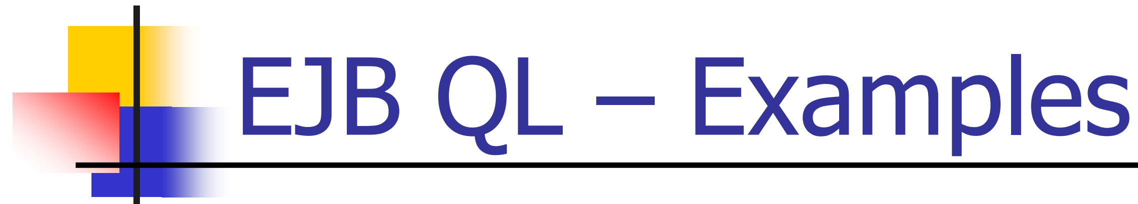
Find all orders that have line items:

```
SELECT OBJECT(o) FROM Order o WHERE o.lineItems IS NOT EMPTY
```

Or…

```
SELECT DISTINCT OBJECT(o) FROM Order o, IN(o.lineItems) item
```

© Copyright 2004, Oracle Corporation



EJB QL – Examples

Find all pending orders:

```
SELECT DISTINCT OBJECT(o) FROM Order o, IN(o.lineItems) item
WHERE item.shipped = FALSE
```

Find all orders for a given product:

```
SELECT DISTINCT OBJECT(o) FROM Order o, IN(o.lineItems) item
WHERE item.product.name = ?1
```

Find names of all products that have been ordered:

```
SELECT DISTINCT item.product.name FROM Order o,
   IN(o.lineItems) item
```

© Copyright 2004, Oracle Corporation

Finders

Query section added to ejb-jar.xml deployment descriptor:

```
<query>
    <query-method>
    <method-name>findOrdersByState</method-name>
    <method-params>
        <method-param>java.lang.Integer</method-param>
    </method-params>
    </query-method>
    <ejb-ql>SELECT OBJECT(order) FROM Order order WHERE
                order.state = ?1</ejb-ql>
</query>
```

© Copyright 2004, Oracle Corporation

Finders

Also for ejbSelects:

```
<query>
    <query-method>
    <method-name>ejbSelectInState</method-name>
        <method-params>
            <method-param>java.lang.Integer</method-param>
        </method-params>
    </query-method>
    <result-type-mapping>Local<result-type-mapping>
    <ejb-ql>SELECT order.orderId FROM Order order WHERE
                order.state = ?1</ejb-ql>
</query>
```

© Copyright 2004, Oracle Corporation

Home Methods

- Method defined on the local or remote home interface
- Abstract bean class implements a method called ejbHome<method>

On EJBEmployeeHome class:

```
public void raiseSalaries(int percent)
    throws FinderException;
```

On abstract EmployeeBean class:

```
public Collection ejbHomeRaiseSalaries(int percent)
    throws FinderException {…}
```

© Copyright 2004, Oracle Corporation

Home Methods

```
public String ejbHomeRaiseSalaries(int percent)
        throws FinderException {
    Iterator employees = ejbSelectAllEmployees().iterator();
    while (employees.hasNext()) {
        Employee emp = (Employee)employees.next();
        int newSal = (int)
                (emp.getSalary() * (1.0+(percent/100.0)));
        emp.setSalary(newSal);
    }
}
```

© Copyright 2004, Oracle Corporation

Relationships

- Allows beans to persistently reference each other (using **object** references instead of PK references!)
- Support for all the major relationship types
  - 1-to-1, Many-to-1, 1-to-Many, Many-to-Many
- Support for bi-directional as well as uni-directional
- May only relate co-located (local) beans
- Must declaratively describe all of the relationships between beans in the deployment descriptor
- Relationships are *automatically* managed by the Container in very well-defined and explicit ways

© Copyright 2004, Oracle Corporation

Agenda

- Setting the Stage
- A New Component Model
- The Entity Requirement
- Updating the Model
- Filling in Holes
- The Golden Age of Simplification
- Summary

© Copyright 2004, Oracle Corporation

EJB 2.1

- Timer Service allows entities to manage higher-level business processes
- Some EJB QL enhancements
- Deployment descriptor uses XSD for validation

© Copyright 2004, Oracle Corporation

Timers

- Meant for coarse-grained business process/workflow events, not real-time events
- Can schedule timeout notifications for:
  - Absolute point in time
  - Elapsed time
  - Recurring time interval
- Beans can create and cancel timers
- Timer callbacks are transactional (as are the create/cancel calls)
- Timers are persistent (survive crashes)

© Copyright 2004, Oracle Corporation

Timers

Entity implements `javax.ejb.TimedObject` interface and can use the timer passed to it in the callback

```
public abstract OrderBean
        implements EntityBean, TimedObject {

    public void ejbActivate() {}
    . . .
    public void ejbTimeout(Timer timer) {
        EventHandler.handleEvent(this.getId(),
                (OrderExpiredEvent)timer.getInfo());
    }
}
```

© Copyright 2004, Oracle Corporation

Timers

- Entity uses TimerService to create a timer and can pass in an object to get associated with the timer:

```
public abstract OrderBean
        implements EntityBean, TimedObject {

    . . .
    public void ejbPostCreate() throws CreateException {
        this.startTimer();
    }

    private void startTimer() {
        this.getEntityContext().getTimerService()
            .createTimer(3600000, new OrderExpiredEvent());
    }
}
```

© Copyright 2004, Oracle Corporation

EJB QL

Ordering and aggregate functions added to EJB QL

- ORDER BY, ASC, DESC

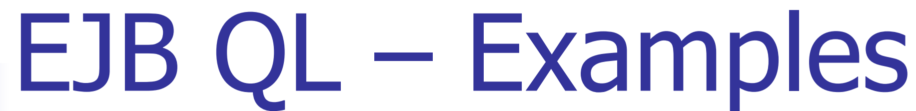
Find orders for 'doodads' sorted by cost:

```
SELECT OBJECT(o) FROM Order o, IN(o.lineItems) item
WHERE item.product.name = 'doodad'
ORDER BY o.totalcost DESC
```

Find total order numbers for 'doodads' sorted by quantity:

```
SELECT item.quantity FROM LineItem item
WHERE item.product.name = 'doodad'
ORDER BY item.quantity ASC
```

© Copyright 2004, Oracle Corporation



EJB QL – Examples

- Aggregate functions:
  - AVG, MAX, MIN, SUM, COUNT

Find the total cost of all of the Orders:

```
SELECT SUM(o.totalCost) FROM Order o
```

Find the highest number of ‘doodads’ ordered:

```
SELECT MAX(item.quantity) FROM LineItem item
WHERE item.product.name = ‘doodad’
```

© Copyright 2004, Oracle Corporation

Agenda

- Setting the Stage
- A New Component Model
- The Entity Requirement
- Updating the Model
- Filling in Holes
- The Golden Age of Simplification
- Summary

© Copyright 2004, Oracle Corporation

EJB 3.0

EJB 3.0 – The "Ease of Use" release

Simplify:
- Simplified API
- Reduce developer artifacts
- Facilitate TDD (test-driven development)

Increase Developer Base:
- Make it accessible to the "average" developer
- Decrease learning curve for new developers

© Copyright 2004, Oracle Corporation

EJB 3.0

- Extensive use of annotations (JSR 175)
- Remove XML deployment descriptor requirement
- Configuration by exception, use defaults to avoid unnecessary declarations
- Eliminate requirement for home interfaces (use an EntityManager)
- Inversion of control techniques (dependency injection)
- Eliminate requirement for component interfaces
- Optionally allow entity to implement POJI (Plain Old Java Interface) interfaces

© Copyright 2004, Oracle Corporation

EJB 3.0

- Enitity is POJO-like concrete class
- Callback (life cycle) methods no longer required
- Detached execution model, side-step the DTO pattern
- Support for entity inheritance and polymorphism
- Many EJB QL enhancements
- Support for native SQL queries
- Dynamic query API
- Scrapped remote entity model
- Standard for object-relational mapping metadata

© Copyright 2004, Oracle Corporation

Annotations

Q: What is the simplest (and most common) model?

A: When the Bean Provider plays all the roles.

Conclusion: Just annotate the bean, stupid!

- Couple the metadata with the bean (where it belongs)
- No XML descriptors required
- Like putting yellow sticky notes on the programming artifacts, instead of creating new ones
- Can be overridden by XML if required

© Copyright 2004, Oracle Corporation

Annotations – Example

```
@Entity
public class Order {
    private Long id;
    ...
    Address shipAddress;
    private Collection<LineItem> lineItems;

    @Id
    public Long getId() { return id; }
    private void setId(Long id) { this.id = id; }
    ...
}
```

© Copyright 2004, Oracle Corporation

Annotations

See paper on OTN about annotations:

http://www.oracle.com/technology/pub/articles/annotations_xml.html

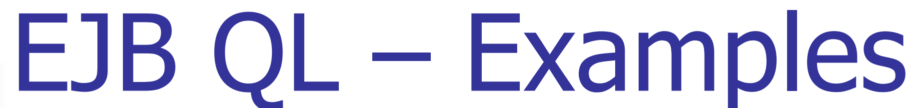
© Copyright 2004, Oracle Corporation

EJB QL

Becoming a more full-bodied query language with many of the missing parts being added:

- Bulk updates/deletes
- Projection lists
- GROUP BY, HAVING
- Subqueries
- More SQL functions

© Copyright 2004, Oracle Corporation



EJB QL – Examples

- Bulk deletes and updates

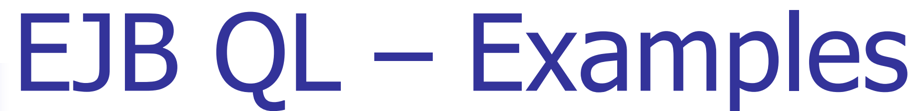
Delete all orders that were billed to 'Bankrupt_R_Us':

```
DELETE FROM Order o
WHERE o.custName = 'Bankrupt_R_Us'
```

Update all orders to be free for 'World Peace Club':

```
UPDATE Order o SET o.totalCost = 0
WHERE o.custName = 'World Peace Club'
```

© Copyright 2004, Oracle Corporation



EJB QL – Examples

- Projection lists

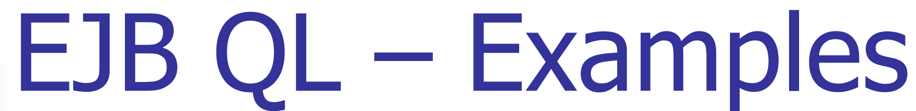
Return the id and cost of all orders over 1000 units of one thing:

```
SELECT o.id, o.totalCost
FROM Order o JOIN o.lineItems item
WHERE item.quantity > 1000
```

Return OrderInfo objects for all orders over a specified cost:

```
SELECT new OrderInfo(o.id, item.product.name, item.cost)
FROM Order o JOIN o.lineItems item
WHERE o.totalCost > :amount
```

© Copyright 2004, Oracle Corporation



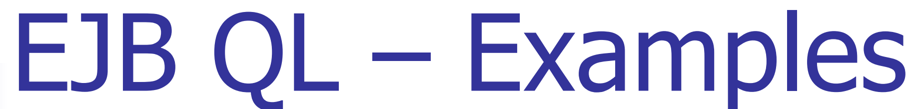
EJB QL – Examples

- GROUP BY, HAVING

Group ‘commercial’ and ‘home’ products by type:

```
SELECT new ProductInfo(
        p.type, p.name, SUM(p.inventory), COUNT(p))
FROM Product p
GROUP BY p.type, p.name
HAVING p.type in (‘commercial', ‘home')
```

© Copyright 2004, Oracle Corporation



EJB QL – Examples

- Subselects

Select employees that have spouses:

```
SELECT DISTINCT emp FROM Employee emp
WHERE EXISTS (
    SELECT spouseEmp FROM Employee spouseEmp
    WHERE spouseEmp = emp.spouse)
```

© Copyright 2004, Oracle Corporation

EJB QL – Examples

- Additional functions including:
  - UPPER, LOWER, TRIM, POSITION
  - CHARACTER_LENGTH, CHAR_LENGTH, BIT_LENGTH
  - CURRENT_TIME, CURRENT_DATE, CURRENT_TIMESTAMP

Select the 'doodad' product:

```
SELECT p FROM Product p
WHERE UPPER(p.name) = 'DOODAD'
```

© Copyright 2004, Oracle Corporation

O-R Metadata

- Direct and relationship mapping types
- Multi-table mappings
- Support for id generation using tables or native DB objects
- Schema definition/generation
- Cascading operations across relationships
- Eager/Lazy loading
- Optimistic locking fields
- Inheritance and discriminators
- Dependent or aggregated objects
- Default settings

EntityManager

```
package javax.ejb;

public interface EntityManager {
   public void create(Object entity);
   public <T> T merge(T entity);
   public void remove(Object entity);
   public Object find(String entityName, Object primaryKey);
   public <T> T find(Class<T> entityClass, Object primaryKey);
   public void flush();
   public Query createQuery(String ejbqlString);
   public Query createNamedQuery(String name);
   public Query createNativeQuery(String sqlString);
   public void refresh(Object entity);
   public void evict(Object entity);
   public boolean contains(Object entity);
}
```

© Copyright 2004, Oracle Corporation

Code Example – Order

```
package examples.ejb30.orderentry;

import javax.ejb.*;

@Entity
@Table(name="ORDER_TAB")
public class Order {

    /** Attributes */
   protected Integer orderId;
   protected int quantity;
   protected String shippingAddress;
   protected Customer customer;
   protected Item item;
```

© Copyright 2004, Oracle Corporation

Code Example – Order

```
/** Accessors */
@Id
public Integer getOrderId() { return orderId; }
public void setOrderId(Integer orderId) { this.orderId = orderId; }

public int getQuantity() { return quantity; }
public void setQuantity(int quantity) { this.quantity = quantity; }

@Column(name="SHIP_ADDR")
public String getShippingAddress() { return shippingAddress; }
public void setShippingAddress(String shippingAddress) {
     this.shippingAddress = shippingAddress;
}
```

© Copyright 2004, Oracle Corporation

Code Example – Order

```
@ManyToOne
@JoinColumn(name="CUST_ID", referencedColumnName="ID")
public Customer getCustomer() { return customer; }
public void setCustomer(Customer customer) { this.customer =
customer; }

@ManyToOne
public Item getItem() { return item; }
public void setItem(Item item) { this.item = item; }

/** Constructors */
public Order() {}

public Order(int quantity, String shippingAddress) {
     setQuantity(quantity);
     setShippingAddress(shippingAddress);
}
}
```

© Copyright 2004, Oracle Corporation

Code Example – Customer

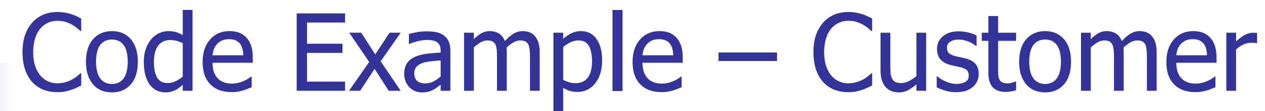
```
package examples.ejb30.orderentry;

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

@NamedQuery(name="findAllCustomersWithName",
   queryString="SELECT c FROM Customer c WHERE c.name LIKE :custName")

@Entity
@Table(name="CUST")
public class Customer {

   /** Attributes */
   protected Integer customerId;
   protected String name;
   protected String city;
   protected Collection<Order> orders;
```

© Copyright 2004, Oracle Corporation

Code Example – Customer

```
/** Accessors */
@Id
public Integer getCustomerId() { return customerId; }
public void setCustomerId(Integer customerId) {
     this.customerId = customerId;
}

public String getName() { return name; }
public void setName(String name) { this.name = name; }

public String getCity() { return city; }
public void setCity(String city) { this.city = city; }

@OneToMany
public Collection<Order> getOrders() { return orders; }
public void setOrders(Collection<Order> orders) {
     this.orders = orders;
}
```

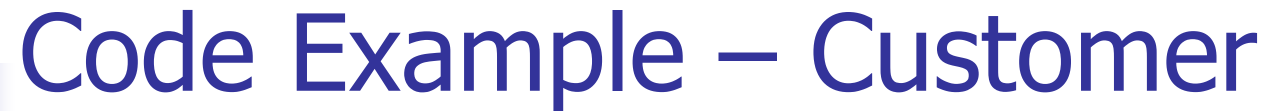
© Copyright 2004, Oracle Corporation



Code Example – Customer

```
@Transient
public String getTransientField() { return "Transient"; }

/** Constructors */
public Customer() {
    orders = new Vector<Order>();
}

public Customer(String name, String city) {
    this();
    this.setName(name);
    this.setCity(city);
}
```

© Copyright 2004, Oracle Corporation



Code Example – Customer

```
    /** Business Methods */
    public void addOrder(Order order) {
         getOrders().add(order);
         order.setCustomer(this);
    }
    public void fillOrder(Order order) {
         getOrders().remove(order);
         order.setCustomer(null);
    }

}
```

© Copyright 2004, Oracle Corporation

Code Example – Item

```
package examples.ejb30.orderentry;

import javax.ejb.*;
import static javax.ejb.AccessType.*;

@Entity
public class Item {

    /** Attributes */
   protected Integer itemId;
   protected String name;
   protected String description;
```

© Copyright 2004, Oracle Corporation

Code Example – Item

```
    /** Accessors */
    @Id
    @Column(name="ITEM_ID")
    public Integer getItemId() { return itemId; }
    public void setItemId(Integer itemId) { this.itemId = itemId; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getDescription() { return description; }
    public void setDescription(String description) {
          this.description = description;
    }
    /** Constructors */
    public Item() {}
    public Item(String name, String description) {
          this.setName(name);
          this.setDescription(description);
    }
}
```

© Copyright 2004, Oracle Corporation

Code Example – Item

```
    /** Business Methods */
    public void addOrder(Order order) {
         getOrders().add(order);
         order.setCustomer(this);
    }
    public void fillOrder(Order order) {
         getOrders().remove(order);
         order.setCustomer(null);
    }

}
```

© Copyright 2004, Oracle Corporation

EntityManager Example

```
@Stateless public class OrderEntry {
   EntityManager em;
   @Inject public void setEntityManager(EntityManager em) {
        this.em = em;
   }
   public void enterOrder(int custID, Order newOrder) {
      Customer cust = (Customer)em.find("Customer", custID);
      cust.getOrders().add(newOrder);
      newOrder.setCustomer(cust);
   }
   public void changedOrder(int custID, Order changedOrder) {
      em.merge(changedOrder);
   }
   public List getAllCustomersNamed(String customerName) {
        return em.createNamedQuery("findAllCustomersWithName")
               .setParameter(1, customerName)
               .listResults();
   }
}
```

© Copyright 2004, Oracle Corporation

Agenda

- Setting the Stage
- A New Component Model
- The Entity Requirement
- Updating the Model
- Filling in Holes
- The Golden Age of Simplification
- Summary

© Copyright 2004, Oracle Corporation

Summary

- ✓ Coarse-grained persistence doesn't cut it for most applications. Persistent objects must be lightweight.
- ✓ Distributed server-side components should access persistent objects, not be them.
- ✓ The majority of apps, by far, talk to relational databases. Live with it, support it and standardize it.
- ✓ Persistence is universal. Persistence frameworks should be easy enough for the universe to use.
- ✓ With enough motivation anything can change!

© Copyright 2004, Oracle Corporation

Summary

- ✓ If you want to succeed, look around and see what the successful commercial people are doing.
- ✓ The EJB 3.0 specification represents the evolution from an inappropriate component model to a modern persistence model based upon the experience of some of the most popular commercial products in the business.
- ✓ J2EE integration and standards-based implementations will ensure EJB's position as the most popular and adopted persistence architecture for the foreseeable future