Enterprise JavaBeans™

Linda DeMichiel

Sun Microsystems, Inc.

Agenda

- Quick introduction to EJB™
- Major new features
  - Support for web services
  - Container-managed persistence
  - Query language
  - Support for messaging
- Status and Roadmap

What is Enterprise JavaBeans™?

An architecture for component-based distributed computing

Part of the Java 2 Platform, Enterprise Edition (J2EE™)

Components written to EJB™ spec can be deployed in any J2EE compatible EJB container without source-code modification or recompilation

Write Once Run Anywhere

EJB Expert Group Work as Part of Java Community Process℠ Program

ATG
BEA
Borland
Fujitsu-Siemens
HP
IBM
IONA
iPlanet
Oracle
Persistence
Pramati
SeeBeyond
Silverstream
Sun
Sybase
Tibco
Webgain
Monson-Haefel

What is an EJB™?

An enterprise bean is a component that contains the business logic that operates on an enterprise's data

EJB components can be

- coarse-grained, remotable
- fined-grained, local

Components run within the EJB Container

EJB Container

Managed environment for the execution of components

Provides platform services to the bean

Container transparently interposes on method invocations to inject its services

Container-Provided Services

Concurrency
Transactions
Distribution
Persistence
Security
Scalability
Resource pooling
EIS Integration
Administration

EJB Component Model

EJB spans different object types:

- Object that represents a conversational session with a client
- Object that represents a stateless service
- Object that represents a web service endpoint
- Object that represents an asynchronously invoked service
- Entity object that represents a business object that can be shared across clients

Component Types

Session Beans

"Conversation with client"

- Stateful
- Stateless

Entity Beans

Model business object as persistent, transactional data, with identity

Message-driven Beans

Asynchronously invoked, anonymous

Parts to an EJB Component

- Client view interface(s)
  - Home interface
  - Component interface
  - Web service endpoint interface
- Bean Class
  - implementation of business logic
- Deployment descriptor
  - declarative specification of Bean's dependencies on operational environment

Home Interface

```
public interface AccountHome extends
  javax.ejb.EJBLocalHome {

    Account create(Customer customer)

      throws CreateException;

    Account findByPrimaryKey(String
  accountID)

      throws FinderException;

...
}
```

Component Interface

```
public interface Account extends
  javax.ejb.EJBLocalObject {

    void debit(double amount)

      throws InsufficientBalanceException;

    void credit(double amount);

    double getBalance();

}
```

Bean Class

```
public class AccountBean implements
  javax.ejb.EntityBean {
    public debit(double amount)
      throws InsufficientBalanceException
     {if (amount > balance)
       throw InsufficientBalanceException;
       else balance = balance - amount;}
      ...
    public double getBalance() {
       return balance;
       }
   ...
}
```

Deployment Descriptor

```
...
<entity>
<ejb-name>Account</ejb-name>
<local-home>com.example.AccountHome</local-hom
  e>
<local>com.example.Account</local>
<ejb-class>com.example.AccountBean</ejb-class>
<persistence-type>Bean</persistence-type>
...
<resource-ref>
<res-ref-name>jdbc/AccountDB</res-ref-name>
<res-type>javax.sql.DataSource</res-type>
</resource-ref>
</entity>
```

Interfaces: Local/Remote

Bean can provide local interface and/or remote interface

- typically not both are provided

Local interfaces new in EJB 2.0

Local EJB interface - standard Java interface

Remote interface - java.rmi interface

Bean Provider needs to consider trade-offs

Local Interface

```
public interface Account extends
  javax.ejb.EJBLocalObject {

    void debit(double amount)

      throws InsufficientBalanceException;

    void credit(double amount);

    double getBalance();

}
```

Remote Interface

```
public interface Account extends
  javax.ejb.EJBObject {
    void debit(double amount)
      throws InsufficientBalanceException,
      RemoteException;
    void credit(double amount)
      throws RemoteException;
    double getBalance()
       throws RemoteException;
}
```

Local vs Remote Trade-offs

Location independence vs more efficient access

Flexibility in distribution vs collocation of components

Loose vs tight coupling between client and bean

Pass-by-value vs "pass-by-reference"

Isolation of components vs ability to share data across components.

Session Beans

- Model stateful service
  - Maintain conversational state
- Model stateless service
- Natural fit for modeling web services

What is a Web Service ?

A set of endpoints operating on messages

Service is described abstractly in WSDL document (XML) and published

Endpoints are defined by set of:

- operations
- messages (arguments, results)

Service can be bound to XML-based protocol (SOAP) and HTTP transport

Providing a Web Service

Create WSDL document describing service

Implement web service endpoints

Publish WSDL

Bottom-up and top-down variants of these approaches are possible

- e.g., Discover WSDL, implement conforming service endpoints

Implementing Web Services with EJB

- Easy! Stateless Session Bean
- Define web service endpoint interface for stateless session bean
- Implement business logic for methods in session bean class
  - Container delegates invocations on service endpoint to session bean instance
  - JAX-RPC runtime handles mapping of requests/responses

Web Service Endpoint Interface

```
public interface StockQuoteProvider
  extends java.rmi.Remote {
    public float getLastTradePrice
       (String tickerSymbol)
      throws java.rmi.RemoteException;
   ...
}
```

Session Bean Class

```
public class StockQuoteProviderBean
  implements javax.ejb.SessionBean {
    public float getLastTradePrice
      (String tickerSymbol)
      throws java.rmi.RemoteException
     {
      // business logic for method;
      ...
      }
}
```

Deployment Descriptor

```
<session>
<ejb-name>StockQuoteEJB</ejb-name>
<service-endpoint>
com.example.StockQuoteProvider
</service-endpoint>
<ejb-class>
com.example.StockQuoteProviderBean
</ejb-class>
<session-type>Stateless</session-type>
<transaction-type>Container</transaction-type>
...
</session>
```

How to Use a Web Service from an Enterprise Bean

- Use much like any other resource
  - use service-ref deployment descriptor element to declare dependency on JAX-RPC service type
- Look up service stub in JNDI
- Get stub/proxy for service endpoint
- Invoke methods on endpoint
- JAX-RPC runtime in container handles invocations on service endpoints

EJB Client View of Web Service

```
public class InvestmentBean implements
  javax.ejb.SessionBean {
public void checkPortfolio(...) {
  Context ctx = new InitialContext();
  StockQuoteService sqs = ctx.lookup(
   "java:comp/env/service/StockQuoteService");
  StockQuoteProvider sqp =
    sqs.getStockQuoteProviderPort();
  float quotePrice sqp.getLastTradePrice(...);
  ...}
}
```

Entity Beans

Model business objects, e.g.,

- Account
- PurchaseOrder
- Employee

Persistent, long-lived entities

Transactional

Queryable

Can have bean-managed or container-managed persistence

Entity Bean Persistence

- Bean Managed Persistence
  - Extremely flexible
  - Can hand-tailor database access
  - Tools can be used to supply data access components
- Container Managed Persistence
  - Frees developer from data access task
  - Allows independence of bean from data source
  - Allows independence of bean from database schema

Container Managed Persistence: Goals

Introduced in EJB 1.0

Completely re-architected in EJB 2.0

Allow for scalable, high-performance implementations

Allow leverage of object-relational mapping technology

Allow wide range of modeling:

- remotable, coarse-grained entities
- fine-grained modeling of persistent state
- relationships among entities to model complex state

Global View

Bean operates in a managed environment

Container provides services to the bean

- Management of persistent state
- Relationship management, including
  - Referential integrity management
  - Collection management
- Query service (for finder methods)

Services have associated contracts/protocols

Deployment descriptor embodies semantic contract between Bean and Container

Bean Provider's View: Abstract Schema

- Logical abstraction over persistent state
- Declaratively defined in deployment descriptor
  - cmp-fields capture persistent state
  - cmr-fields capture persistent relationships
- Embodied in method-based API
  - abstract get and set methods defined for access to persistent state and relationships
  - java.util.Collection API for collection-valued cmr-fields
- Provides basis for declarative query language

Container's View

Provides implementation of abstract schema

Provides state management

Provides relationship management

Provides implementation of declarative queries against abstract schema

Spec allows wide variety of implementation techniques

Container-Managed Support for Relationships

1-1, 1-N, M-N associations among beans

Provides programmatic navigability

Container maintains referential integrity

Defined by Bean Provider in deployment descriptor

Tightly-integrated set of beans: assumes co-location in same JVM

Example

```
public abstract class OrderBean implements
  javax.ejb.EntityBean {
 public abstract java.util.Collection
   getLineItems();
 ...
 public void addLineItem(Product p, int
  quantity) {
    LineItemHome = .../JNDI lookup
    LineItem l = LineItemHome.create();
    l.setQuantity(quantity);
    l.setProduct(p);
    getLineItems().add(l);
}...}
```

Expected Design Patterns

Use of session beans and message-driven beans to front network of entity beans

Use of remote and/or coarse-grained entity beans as aggregators for internal network of fine-grained entity beans

Expect dual-mode remote+local interface use to be relatively uncommon

EJB™ QL

Portable definition of query methods for container-managed persistence entities

New in EJB 2.0

Declarative language, independent of data store

Queries defined at abstract schema level in deployment descriptor

SQL-like SELECT...FROM...WHERE syntax

Based on navigability over relationships

Supports parameterized queries

Query Methods

- Finder methods
  - Return EJBObjects or EJBLocalObjects of same type as entity bean
- Select methods
  - for internal use of bean class
  - can return any cmp- or cmr-field type
  - fuller range (superset) of queries available

EJB QL Example

Find orders for a specific product:

```
SELECT OBJECT(o)
 FROM Orders o, IN(o.lineItems) l
 WHERE l.product.name = ?1
```

EJB™ QL Example

Order by quantity and cost:

```
SELECT OBJECT(o)
 FROM Customer c, IN(c.orders) o
 WHERE c.address.state = 'CA'
ORDER BY o.quantity ASC,
         o.totalcost DESC
```

EJB™ QL Example

Use of aggregate function:

```
SELECT SUM(l.price)
 FROM Order o, IN(o.lineItems) l
 WHERE o.customer.lastname = 'Smith'
AND o.customer.firstname = 'John'
```

Message-Driven Beans

- Provide loose coupling among heterogeneous systems
  - Interoperate with legacy systems that use messaging for integration
  - Interoperate with B2B systems not based on the J2EE platform
- Provide asynchronous communication
- Provide support for disconnected use of enterprise beans

Message-Driven Beans

New enterprise bean type added in EJB 2.0

- Asynchronous
- Activated upon message arrival
- Stateless
- No home or component interface
- Message listener method in message-driven bean class contains business logic
- Use with container-managed or bean-managed transaction demarcation

Message-Driven Bean Enhancements

Message-driven beans were very JMS-centric in EJB 2.0

Generalizing in EJB 2.1 to support other messaging types

e.g., JAXM

Message-driven bean class can implement specific messaging interface

- Previously restricted to javax.jms.MessageListener

Pluggability of messaging providers through Connector APIs in J2EE 1.4

EJB 2.0

Main new features:

- Message-driven beans
- Container-managed persistence
- EJB QL
- Interoperability contracts
- Security Enhancements

Released as part of J2EE™ 1.3 platform in September 2001

- Specification
- Reference Implementation
- Compatibility Test Suite
- http://java.sun.com/products/ejb/docs.html

EJB 2.1

Main new features:

- Web service support
- EJB QL enhancements
- Message-driven bean extensions
- Timer service

EJB 2.1 currently in JCP Community Review

Public draft targeted for late June

For More Information

http://java.sun.com/products/ejb

http://java.sun.com/j2ee