TDDD05

Enterprise Java Beans – Part 2
Lecture 08
(slides from Mikhail Chalabine)

Lu Li lu.li@liu.se
Department of Computer and Information Science
Linköping University, Sweden

Some slides from Mikhail Chalabine and Peter Nunus

EJB 2 Final thoughts

- **Not object-oriented**
  - Data and operations separated
    - Entity beans encapsulate data (DB record-oriented)
    - Session beans encapsulate functionality (no data)
  - No component inheritance (EJB 2.0)
    - EJB 3.0 – Beans are POJOs
      - Component inheritance implemented as classical OO inheritance
- **Development and Application**
  - Strict architecture
  - Complex
  - Beans are difficult to test
    - Container required
    - Deployment errors mistaken for Business Logic errors

EJB

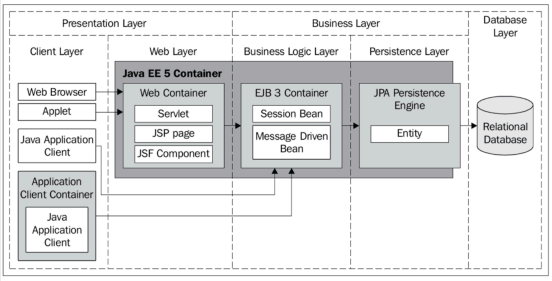
EJB 3

EJB 3

- **EJB 3 – POJO configurable through metadata annotations**
  - **Remote interface**
    - Bean's business methods
    - Amenable to Reflection metadata analysis
  - **Bean class**
    - Business logic

[EJB 3 In Action, 2007, p.8] EJB

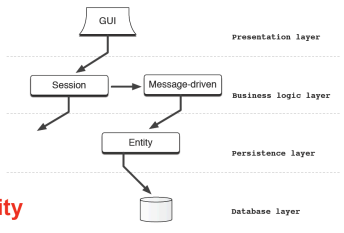
EJB 3 Architecture



[EJB 3 Developer Guide, 2008, p.8] EJB

EJB 3 Bean Types

- **Session**
  - **Stateless**
  - **Stateful**
- **Message-Driven**



- **Entity = JPA Entity**

EJB

EJB 3 Entities (1)

- **Java Persistent API Entities (1)**
  - **Not EJB Beans**
    - **Not distributed objects**, *i.e.*, not EJB components
      - Have **no access** to Container services, e.g., transactions, etc.
      - **Clients must always be local to JVM**
        - No remote access
      - Mitigated by Façade design pattern
    - **Can run outside EJB container**
      - EJB Container
      - Web Container
      - Java SE Applications

EJB 3 Entities (2)

- **Java Persistent API Entities (2)**
  - **POJOs**
    - Fully **Object-oriented**
      - Can encapsulate both data and operations
      - ***Cf.*** EJB 2 Entity Beans
        - Data carriers (a.k.a. DAOs)
        - No business logic beyond getters/setters and data validation
    - Standard way of doing **Object-relational mapping**

EJB 3 Entities (3)

- **Java Persistent API Entities (3)**
  - Managed by Entity Manager
  - ***persistence.xml***
    - Describes Persistence Module
    - Defines Persistence Units

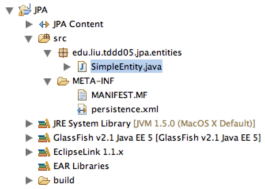
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">

<persistence-unit name="TDDD05" transaction-type="JTA">
      <provider>oracle.toplink.essentials.ejb.cmp3.EntityManagerFactoryProvider</provider>
      <jta-data-source>jdbc/mySQL</jta-data-source>
      <class>edu.liu.tddd05.jpa.entities.SimpleEntity</class>
    </persistence-unit>
</persistence>
```


EJB 3 Entities (4)

```
@Entity
@Table(name="t_person")
public class SimpleEntity implements Serializable {


      @Id
      private int id;
      private String fName;
      private String lName;

      public SimpleEntity() {
            super();
      }
      public int getId() {
            return this.id;
      }

      public void setId(int id) {
            this.id = id;
      }

      ...
}
```


EJB 3 Entities (5)

- **JPA**
  - Metadata-driven POJO technology
  - **Persistence Unit** – a group of entities packaged in an application module
    - Requires JPA engine
      - GlassFish – Oracle TopLink
      - toplink-essentials.jar



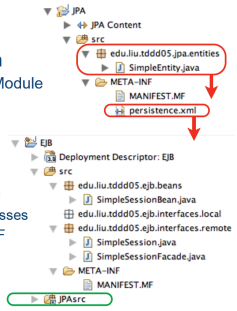
EJB 3 Entities (6)

- **Entity Manager API**
  - Most important API in JPA
  - Manages lifecycle of entities
  - Bridge between OO and relational worlds
    - Translates Entities into table records and back

```
@PersistenceContext(unitName="TDDD05")
EntityManager em;

public String getIsbn( int id )
{
      SimpleEntity eb = em.find( SimpleEntity.class, id );
      return eb.getISBN();
}
```

EJB 3 Entities (7)

- **Packaging**
  - **Entity in Java EE Application**
    - Package in a standard Java EE Module
      - JAR or WAR
    - Place into target EAR
      - Root or Library directory
  - **Entity in EJB Module**
    - Merge JPA code into EJB module
      - Put entity together with Bean classes
      - Put *persistence.xml* in *META-INF*
      - Package as a single EJB JAR



EJB 3 Remote Interface (1)

- **Similar to EJB 2**
  - Acts as proxy
  - Defines business methods available to clients
- **New Features**
  - **New** annotation-based **declaration**
  - **New retrieval protocol** [See also <<Container Services>> section below]
    - No need to use EJBHome factory
    - Classical JNDI lookup **or** **Dependency Injection (DI)**

```
import javax.ejb.Remote;

@Remote
public interface SimpleSession
{
    public String sayHello();
}
```


Client with DI EJB 3 Remote Interface (2)

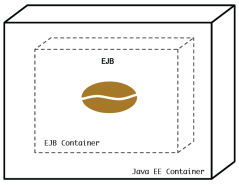
No reference to EJBHome, Implicit lifecycle management, *e.g.*, no call to `create()`

```
import javax.ejb.EJB;

public class SessionBeanClient
{
    @EJB
    private static SimpleSession simpleSession;

    private void invokeSessionBeanMethods()
    {
        System.out.println( simpleSession.sayHello( “John” ));
        System.out.println( "\nSimpleSession is of type: "
                            + simpleSession.getClass().getName());
    }

    public static void main( String[] args )
    {
        new SessionBeanClient().invokeSessionBeanMethods();
    }
}
```


Container Services

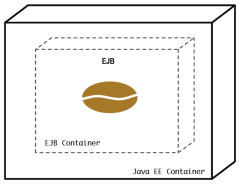
EJB & Java EE Containers

- Process client requests
  - Generates stubs and skeletons
  - Creates EJB instances as needed
  - Persists entity beans (EJB 2)
  - Handles security and transactions
- Provide **middleware services**



EJB & Java EE Containers

- Process client requests
  - Generates stubs and skeletons
  - Creates EJB instances as needed
  - Persists entity beans (EJB 2)
  - Handles security and transactions
- Provide **middleware services**



Middleware

- **Software that connects components and applications**
  - **Explicit middleware** (**CORBA**) :
    - Programmer writes to API
    - Difficult to develop, maintain, support
  - **Implicit middleware** (**EJB**)
    - Programmer writes business logic
    - Declarative middleware specification
    - Middleware generated automatically

EJB & Java EE Container Services

- A.k.a. **EJB middleware services**:
  - Remotability (RMI/IIOP)
  - Transactions (JTA)
  - Logging
  - Resource pooling
  - Client state management
  - Messaging (JMS)
  - Interceptors (AOP)
  - Integration (DI)
  - Timers
  - Security
  - Thread safety
  - Integration (DI)
  - Persistence (JPA)
  - Caching

Business logic layer
Persistence layer

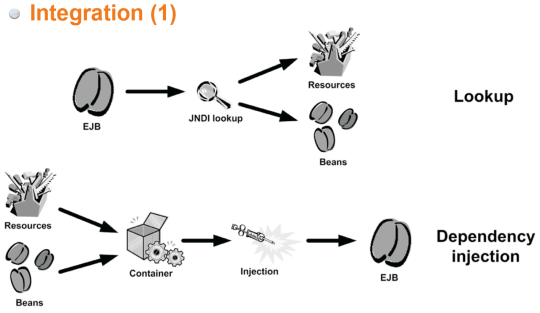
EJB & Java EE Container Services

- A.k.a. EJB middleware services:
  - **Remote Method Invocation** (RMI-IIOP)
  - Transactions (JTA)
  - **Persistence** (JPA)
  - Resource pooling
  - State management
  - Messaging (JMS)
  - **Integration** (DI)
  - Interceptors (AOP)
  - Authentication
  - Authorization

Lab 2

EJB & Java EE Container Services

- **Integration (1)**



EJB & Java EE Container Services

- **Integration (2)**
  - **Reminds "early binding" but is not one**
  - Reminder
    - **Early binding**
      - Static linking of objects – compile-time linking
    - **Late binding**
      - Run-time reference resolution a.k.a. dynamic loading
  - **Integration is run-time activity**
    - Java resolves variable references and method calls dynamically
    - Run-time exceptions if fails

EJB & Java EE Container Services

- **Integration (3)** **JNDI Lookup example**

```
public class SimpleClient {
    ...

    public void sayHelloViaEjb() throws Exception {

        Properties     p      = new Properties();

        p.put( "org.omg.CORBA.ORBInitialHost", "127.0.0.1" );
        p.put( "org.omg.CORBA.ORBInitialPort", "3700" );

        InitialContext ctx    = new InitialContext( p );
        SimpleSession  bean   = (SimpleSession) ctx.lookup(
                                              "SimpleSession" );

        String         result = bean.sayHello( "John" );
    }
    ...
}
```

EJB & Java EE Container Services

- **Integration (4)** Dependency Injection example (1)

```
public class SimpleClient {
    ...

    @EJB( mappedName="corbaname:iiop:127.0.0.1:3700#SimpleSession" )
    public static SimpleSession beanDi;

    public void useEJB() throws Exception {

        String result = beanDi.sayHello( "John" );

    }
    ...
}
```


EJB & Java EE Container Services

- **Integration (5)** Dependency Injection example (2)

```
@Stateless
public class BookSessionBean implements BookSession {
     @Resource( name="jdbc/MySQL" )
     private DataSource dataSource;

     public Integer gettingQuantity( String bookId ) {
          Integer qnty;
          try {
               Connection        conn   = dataSource.getConnection();
               PreparedStatement stmt   = conn.prepareStatement(
                                          "SELECT quantity FROM books WHERE isbn= ?");
               stmt.setString( 1, bookId );
               ResultSet         rslt   = stmt.executeQuery();
               rslt.next();
               qnty                     = rslt.getInt("quantity");
               rslt.close();
               stmt.close();
               conn.close();
          } catch (Exception e) { throw new EJBException( e.getMessage() ); }
          return qnty;
     }
}
```

[Beginning Database-Driven Application Development in Java EE Using GlassFish, Vasiliev, 2008, p. 154]

EJB & Java EE Container Services

- **Integration (5)** Dependency Injection example (2)

```
@Stateless
public class BookSessionBean implements BookSession {
     @Resource( name="jdbc/MySQL" )
     private DataSource dataSource;

     public Integer gettingQuantity( String bookId ) {
          Integer qnty;
          try {
               Connection        conn   = dataSource.getConnection();
               PreparedStatement stmt   = conn.prepareStatement(
                                          "SELECT quantity FROM books WHERE isbn= ?");
               stmt.setString( 1, bookId );
               ResultSet         rslt   = stmt.executeQuery();
               rslt.next();
               qnty                     = rslt.getInt("quantity");
               rslt.close();
               stmt.close();
               conn.close();
          } catch (Exception e) { throw new EJBException( e.getMessage() ); }
          return qnty;
     }
}
```

[Beginning Database-Driven Application Development in Java EE Using GlassFish, Vasiliev, 2008, p. 154]

EJB & Java EE Container Services

- **Integration (6)**
  - **Dependency Injection**
    - Container injects resources using Java Metadata Annotations
      - EJBs via `@EJB`
      - Persistence Context via `@PersistenceContext`
      - Persistence Unit via `@PersistenceUnit`
      - Other resources via `@Resource`
  - **Applies**
    - **Session Beans**
    - **Message-driven Beans**

EJB & Java EE Container Services

- **Integration (5)** Dependency Injection example (2)

```
@Stateless
public class BookSessionBean implements BookSession {
     @Resource( name="jdbc/MySQL" )
     private DataSource dataSource;

     public Integer gettingQuantity( String bookId ) {
          Integer qnty;
          try {
               Connection        conn   = dataSource.getConnection();
               PreparedStatement stmt   = conn.prepareStatement(
                                          "SELECT quantity FROM books WHERE isbn= ?");
               stmt.setString( 1, bookId );
               ResultSet         rslt   = stmt.executeQuery();
               rslt.next();
               qnty                     = rslt.getInt("quantity");
               rslt.close();
               stmt.close();
               conn.close();
          } catch (Exception e) { throw new EJBException( e.getMessage() ); }
          return qnty;
     }
}
```

[Beginning Database-Driven Application Development in Java EE Using GlassFish, Vasiliev, 2008, p. 154]

EJB & Java EE Container Services

- **Integration (6)**
  - **Dependency Injection**
    - Container injects resources using Java Metadata Annotations
      - EJBs via `@EJB`
      - Persistence Context via `@PersistenceContext`
      - Persistence Unit via `@PersistenceUnit`
      - Other resources via `@Resource`
  - **Applies**
    - **Session Beans**
    - **Message-driven Beans**

EJB & Java EE Container Services

- **Integration (7)**
  - **Dependency Injection**
    - Limitations
      - **No Container – No Injection!**
      - **Once Injected – Always Injected!**
        - Configuration fixed
          - Can be done at runtime (e.g., JNDI access needed) but
          - No procedural composition `if (...) then @Resource`
      - Can't inject POJOs, only properly declared EJBs
        - See **Spring** Framework and AOP if you need it

EJB & Java EE Container Services

- **Integration (8)**
  - **Dependency Injection**
    - Glues components together
    - Declarative configuration instead of programmable lookups
  - **No, No!**
    - Inject Stateful EJB into a Stateless EJB

```
@Stateless
public class Ejb2 implements I2 {
   ...
   @EJB
   private Ejb1 ejb1;
   ...
}
```

```
@Stateful
public class Ejb1 implements I1 {
   ...
}
```

EJB & Java EE Container Services

- **Remotability**
  - Remote Method Invocation over Internet Inter-Orb Protocol (**RMI / IIOP**)
  - Enables **remote access** to EJBs
  - **Applies**
    - **Session Beans**
  - **Location Transparency**
    - Invoke **Local** and **Remote EJBs** same way
      - Local – deployed and executed in the same JVM
      - `@Local`, `@Remote`
    - **Best practices**
      - Define EJBs as Local whenever possible

EJB & Java EE Container Services

- **Transactions**
  - Sequence of one or more steps that add, modify, or delete persistent data
  - ACID
  - Java Transaction API (JTA)
  - Declarative configuration
  - Any method can be transactional
  - Container commits if completes properly
  - **Applies**
    - **Session Beans**
    - **MDBs**

EJB & Java EE Container Services

- **Logging**
  - Process by which Enterprise Server captures data about events that occur during it's operation
  - Records in **log file**
  - Implementation
    - **Vendor specific** to a standard
    - Alternative: Apache Commons Logging Library
  - **Obs!** [JSR 220: EJB, Version 3.0, Sec. 21.1.2 Programming Restrictions, p. 545]
    - EJB must not use **AWT functionality** to output to a display, or to input from a keyboard
    - EJB must not use `java.io` **package functionality** to access files and directories in the file system
      - Use resource manager API, such as JDBC, to store data

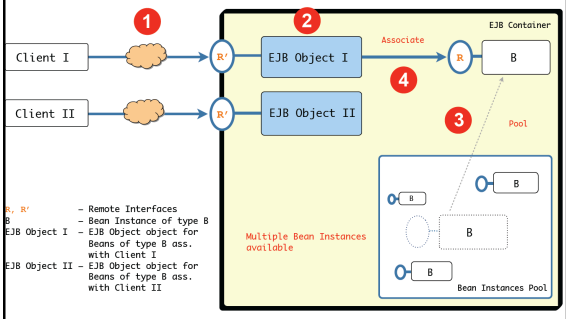
EJB & Java EE Container Services

- **Resource pooling and Management (1)**
  - **Instance Pooling**
  - **Passivation / Activation**

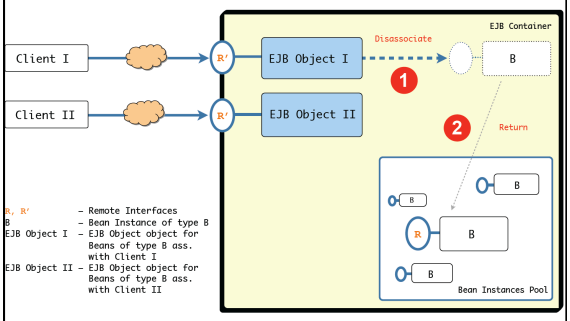
EJB & Java EE Container Services

- Resource pooling and Management (2)
  - Instance Pooling
    - Applies
      - Stateless Session Beans
      - MDBs
    - ∀ EJB ∃ a pool of component instances
    - Goal
      - Reuse and Resource optimization
        - Bean instances
        - Instance returns to pool after serving a client
        - Faster than garbage collection

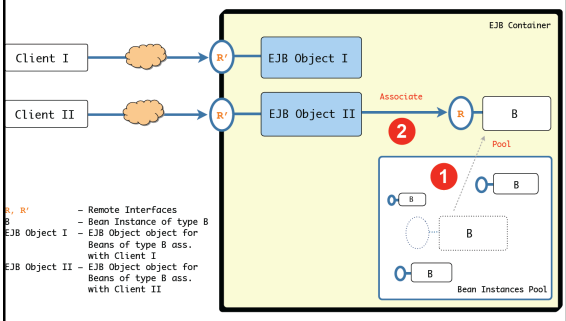
Pooling and Reuse - Stateless (1)



Pooling and Reuse - Stateless (2)



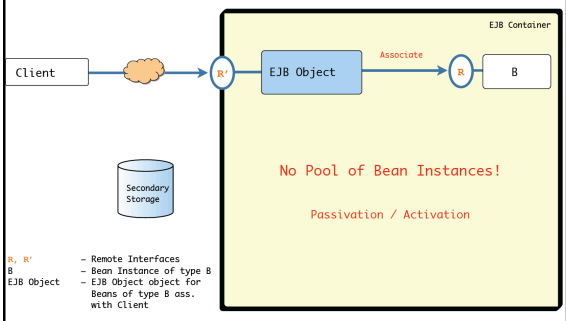
Pooling and Reuse - Stateless (3)



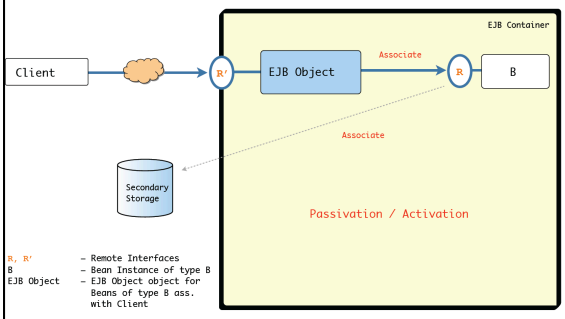
EJB & Java EE Container Services

- Resource pooling and Management (3)
  - Passivation / Activation
    - Applies
      - Stateful Session Beans

Pooling and Reuse - Stateful (1)

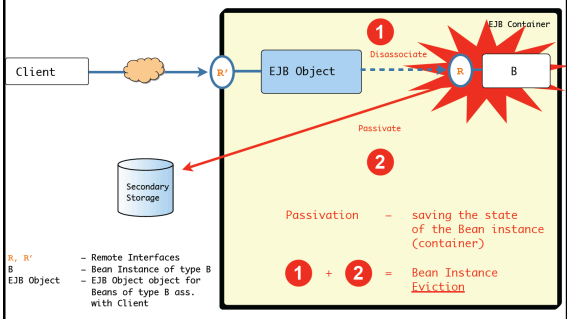


Pooling and Reuse - Stateful (2)



EJB

Pooling and Reuse - Stateful (3)



EJB

EJB & Java EE Container Services

- State management
  - Applies
    - Stateful Session Beans
  - Container maintains conversational state
  - Container maintains session
  - Programmer works with instance variables

EJB

EJB & Java EE Container Services

- Messaging
  - Applies
    - Message-driven Beans
  - Container supports JMS
  - Programmer can abstract from implementation details

EJB

EJB & Java EE Container Services

- Interceptors (1)
  - Objects automatically triggered when an EJB method is invoked
  - Applies
    - Session Beans
    - Message-driven Beans
  - Interceptor
    - Regular Java class with annotated methods
    - Attached to either EJB class or method
    - Triggered when EJB method execution begins
    - Can complete after EJB method returns
    - Can inspect whatever EJB method returns
    - In AOP terms – an around invoke advice
    - `@AroundInvoke`, `@PostConstruct`, `@PrePassivate`, `@PostActivate`, `@PreDestroy`

[EJB 3 in Action, 2007, pp. 157] EJB

EJB & Java EE Container Services

- Interceptors (2)

```
public class ActionLogger {
    @AroundInvoke
    public Object logMethodEntry( InvocationContext invocationContext ) throws Exception {
        System.out.println( "Entering method: " + invocationContext.getMethod().getName() );
        return invocationContext.proceed();
    }
}

@Stateless
public class MyEJB {
  ...
  @Interceptors( ActionLogger.class )
  public void foo() {
    ...
  }
}
```

[EJB 3 in Action, 2007, pp. 157] EJB

EJB & Java EE Container Services

- **Timers**
  - Time-delayed callbacks
  - Timeout method invoked after time interval elapses
  - Persistent – can survive a crash
  - Transactional – a transaction failure in a timeout method rolls back the actions taken by the timer

```
public class MyEJB {
     ...
     @Resource TimerService timerService;                          Inject Timer service
     ...
     public void addRecord( Record record ) {
          ...
          timerService.createTimer( 15*60*1000, 15*60*1000, record);   Create a timer
          ...
     }
     ...
     @Timeout
     public void monitorRecord( Timer timer ) {                    Timeout method
          Record record = ( Record ) timer.getInfo();
          ...
     }
}
```


EJB & Java EE Container Services

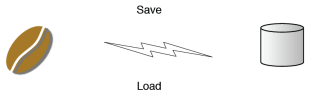
- **Security**
  - Authentication
  - Authorization
  - **Applies**
    - **Session Beans**
  - **Container integrates with JAAS**
    - Alike JPA entities external to EJBs
  - Simple configuration instead of complex coding
    - Roles
    - Realms

EJB & Java EE Container Services

- **Thread safety**
  - **Applies**
    - **Session Beans**
    - **Message-driven Beans**
  - **Container makes components tread-safe**
  - **Container optimizes performance**
  - Programmer writes beans to single-threaded paradigm

EJB & Java EE Container Services

- **Persistence**
  - **Applies**
    - **JPA Entities**
  - Process of preserving consistency between EJB properties and Database tuples
  - Declarative mapping
  - Declarative configuration
  - Alternative to low-level JDBC/SQL



EJB & Java EE Container Services

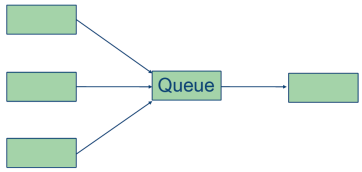
- **Caching**
  - **Applies**
    - **JPA Entities**
  - **Container provides**
    - Efficient data caching
    - Performance optimizations
    - Application tuning

How container vendors compete?

- Better and more efficient container services
- Caching strategies
- Development tool integration
- Database access optimization
- Performance

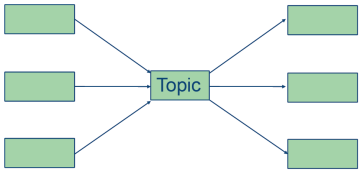
EJB Patterns

Point-to-Point



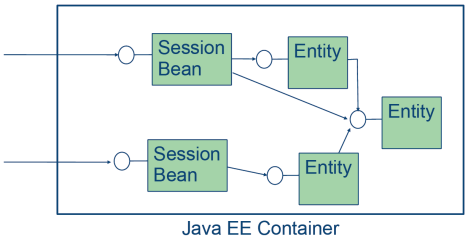
EJB

Publish - Subscribe



EJB

Façade



EJB

EJB Generations Comparison

EJB 2 vs. EJB 1.1

- **EJB 2 advantages (1)**
  - **Local interfaces**
    - Can declare Beans as local (same JVM). No need to wrap as remote. Arguments passed by reference, not by value. Improves performance.
  - **ejbHome methods**
    - Entity beans can declare ejbHomeXXX(…) methods that perform operations related to the EJB component but that are not specific to a bean instance. The ejbHomeXXX(…) method declared in the bean class must have a matching home method XXXX(…) in the home interface.

EJB

EJB 2 vs. EJB 1.1

- **EJB 2 advantages (2)**
  - **Message Driven Beans (MDB)**
    - New enterprise bean type to handles incoming JMS messages
  - **New Container Managed Persistence (CMP) Model**
    - New contract called the abstract persistence schema. The container handles persistence automatically at runtime.
  - **EJB Query Language (EJB QL)**
    - SQL-based language that allows to implement and execute finder methods. EJB QL also used in new query methods ejbSelectXXX(…), which is similar to ejbFindXXX(…) methods except that it is only for the bean class to use and not exposed to the client (i.e. it is not declared in the home interface)

EJB 2 vs. EJB 1.1

- **EJB 2 advantages (3)**
  - **Container-managed timer service**
    - Provides coarse-grained, transactional, time-based event notifications enabling enterprise beans to model and manage higher-level business processes.
  - **Web Service support**
    - EJB 2.1 adds the ability of stateless session beans to implement a Web Service endpoint via a Web Service endpoint interface.
  - **Enhanced EJB-QL**
    - Includes support for aggregate functions and ordering of results.

EJB 2 vs. EJB 1.1

- **EJB 2 disadvantages**
  - **Must create** certain component interfaces
  - **Must implement** certain call-back methods
  - **EJB deployment** descriptors are complex
  - **EJB** components are **not** truly **object oriented**
    - Use of inheritance and polymorphism restricted
  - **EJB** modules **cannot be tested** outside EJB container
    - Debugging inside container difficult

› p. 169 Lulu.com Java J2EE Job Interview Companion 2nd Edition Apr 2007.

EJB 3 vs. EJB 2

- Simplified Persistence API (JPA)
- Metadata annotations vs deployment descriptors
- Improved Query Language
- Use of Defaulting
- Dependency Injection
- Simplification of Session Beans

EJB 3 vs. EJB 2

- **Simplified EJB implementation cycle**
  - EJB 3.0 eliminates the need for **home** and **component interfaces** and the requirement for bean classes for **implementing javax.ejb.EnterpriseBean** interfaces.
  - EJB bean class is a simple Java class – **POJO.**
  - EJB interface is a simple business interface.

EJB 3 vs. EJB 2

- **Use of Java Annotations Instead of XML-based Deployment Descriptors**
  - Metadata annotation as an alternative to deployment descriptors.
  - Annotations can be used to specify bean types, different attributes such as transaction or security settings, O-R mapping and injection of environment or resource references.
  - Deployment descriptor can be used to override metadata annotations.

EJB 3 vs. EJB 2

- **Interceptor capabilities**
  - An interceptor is a method that intercepts a **business method** invocation.
  - An interceptor method may be defined in a Stateless Session Bean, Stateful Session Bean or an interceptor class may also be used instead of defining the interceptor method in the bean class.

EJB 3 vs. EJB 2

- **Simplified JNDI lookup of EJB**
  - Lookup of EJB has been simplified and clients do not have to create a bean instance by invoking create method on EJB and can directly invoke a method on the EJB.
  - DI FIXME! EJB 3 In Action 2007, p. 40, 22, 26

EJB Best Practices

Best Practices

- **Session Beans**
  - Choose Stateless Session Bean
    - Container will never persist stateless session bean
  - Choose Local interface over Remote if same JVM
    - Remote Interface implies network access – expensive
  - Choose HttpSession over EJB Stateful Beans
    - Can do if Client is a Web-tier Client
  - Separate crosscutting concerns
    - Place Logging and auditing in Interceptors (see below)
  - See EJB 3 In Action 2007, p. 108

References

- http://java.sun.com/products/ejb/docs.html

Resources

- Szyperski, chapter 14
- TDDD05 course page with links to EJB tutorials and books
- JBoss, Open source EJB Container
  - http://www.jboss.org
- Sun Microsystems. EJB 3.0 Specification Final Release
- Sun Microsystems. The Java EE 5 Tutorial, 2006