Implementing Java EE Applications Using Enterprise JavaBeans (EJB) 3 Technology: Real World Tips, Tricks, and New Design Patterns

Fabiane Bizinella Nardon

CTO
Zilics

www.zilics.com

Edgar Silva

Solutions Architect
JBoss (Red Hat)

www.jboss.com

Session TS-4721

Goal

Share the tips, tricks, and new design patterns we learned when developing real-world Enterprise JavaBeans™ (EJB™) 3 technology applications.

Agenda

Brief Introduction to EJB 3 Technology New Features

EJB 3 Technology in the Real World

Session Beans Pitfalls, Tips, and Tricks

Entity Beans Pitfalls, Tips, and Tricks

Message Driven Beans Pitfalls, Tips, and Tricks

Refactoring to Better Use EJB 3 Technology

Old and New Design Patterns

Conclusion

Agenda

Brief Introduction to EJB 3 Technology New Features

EJB 3 Technology in the Real World

Session Beans Pitfalls, Tips, and Tricks

Entity Beans Pitfalls, Tips, and Tricks

Message Driven Beans Pitfalls, Tips, and Tricks

Refactoring to Better Use EJB 3 Technology

Old and New Design Patterns

Conclusion

EJB 3 Technology New Features

- POJO based
- Annotations support
- Callback methods
- Listeners
- Interceptors
- Dependency Injection
- Defaults
- New persistence model
- New query language

Agenda

Brief Introduction to EJB 3 Technology New Features

EJB 3 Technology in the Real World

Session Beans Pitfalls, Tips, and Tricks

Entity Beans Pitfalls, Tips, and Tricks

Message Driven Beans Pitfalls, Tips, and Tricks

Refactoring to Better Use EJB 3 Technology

Old and New Design Patterns

Conclusion

EJB 3 Technology in the Real World

- No more Hello World or Pet Store applications, please!
- Some real-world projects

Project: [redacted]

Healthcare Information Systems for Healthcare Providers in Brazil and Angola

Project: [redacted]

Enterprise Content Management for a Government Agency

Project: [redacted]

Financial Services Sales System for a multi-national bank

Project: [redacted]

Human Resources Information system for an IT Company

Agenda

Brief Introduction to EJB 3 Technology New Features

EJB 3 Technology in the Real World

Session Beans Pitfalls, Tips, and Tricks

Entity Beans Pitfalls, Tips, and Tricks

Message Driven Beans Pitfalls, Tips, and Tricks

Refactoring to Better Use EJB 3 Technology

Old and New Design Patterns

Conclusion

Keep the Defaults in Mind!

Session beans pitfalls

- Watch out for default transaction demarcation: by default, all methods are transactional with a REQUIRED transaction attribute
- What to do with non-transactional methods?
  - Use NOT_SUPPORTED: the method won't have access to the transaction context
  - Use SUPPORTS: take care, the behavior can be different depending on the calling method
  - Use NEVER: can cause an error if the calling method is transactional

Agenda

Brief Introduction to EJB 3 Technology New Features
EJB 3 Technology in the Real World
Session Beans Pitfalls, Tips, and Tricks
Entity Beans Pitfalls, Tips, and Tricks
Message Driven Beans Pitfalls, Tips, and Tricks
Refactoring to Better Use EJB 3 Technology
Old and New Design Patterns
Conclusion

Lazy Loading Relationships and Tiers

- You should annotate your relationship properties as LAZY in most cases, “Why?”
  - You won’t have long queries, retrieving unnecessary objects, resulting on bad performance
- But there are situations when it is better to use EAGER loading
  - It is better to eager load the relationships your view tier is expecting, so you can avoid multiple database queries
  - If you are not keeping the session opened, you have to initialize your relationships before passing the object to the view tier to avoid lazy loading exceptions

Detached Entities and Lazy Loading

```
public Patient getPatientData(PK id) {
   Patient p = entityManager.find(id);
   p.getDocuments().size();
   // Objects detach automatically when they
   // are serialized or when a persistence context ends.
   // The specification does not define any way to
   // explicitly detach objects.
}

public void updatePatient(Patient p) {
    Patient p = entityManager.merge(p);
}
```

Retrieving Meta-Information

- You can read annotation in run-time to retrieve information about your model
- However, if a deployment descriptor was used, there is only one safe way to do this

```
public String getPrimaryKey(Class clazz, EntityManager em) {
    if (em instanceof HibernateEntityManager) {
        SessionFactory factory = ((HibernateEntityManager)em)
            .getSession().getSessionFactory();
        String pk = factory.getClassMetadata(clazz)
                        .getIdentifierPropertyName();
    }
}
```

Agenda

Brief Introduction to EJB 3 Technology New Features

EJB 3 Technology in the Real World

Session Beans Pitfalls, Tips, and Tricks

Entity Beans Pitfalls, Tips, and Tricks

Message Driven Beans Pitfalls, Tips, and Tricks

Refactoring to Better Use EJB 3 Technology

Old and New Design Patterns

Conclusion

Message Driven Beans Pitfalls

- Dealing with Exceptions
  - The Spec says: “Message-driven beans should not, in general, throw RuntimeExceptions”
  - How to rollback a transaction
    - Throw an Exception annotated with “rollback=true”
    - Catch the exception and call setRollbackOnly()
- Missed PreDestroy Callbacks
  - You can not assume that the PreDestroy callback method will always be invoked; It may not be executed when
    - The EJB technology Container crashes
    - A system exception is thrown from the instance’s method to the container

Agenda

Brief Introduction to EJB 3 Technology New Features

EJB 3 Technology in the Real World

Session Beans Pitfalls, Tips, and Tricks

Entity Beans Pitfalls, Tips, and Tricks

Message Driven Beans Pitfalls, Tips, and Tricks

Refactoring to Better Use EJB 3 Technology

Old and New Design Patterns

Conclusion

Using Dependency Injection

- You can dynamically get resources using DI, such as Connections, Persistence Contexts, Queues and others EJB technology
- Dependency Injection in EJB 3 technology is as easy as on Spring, even for dynamic values (env-entries)

```
@Stateless
public class SalesProcessorBean implements
           SalesProcessorRemote, SalesProcessorLocal {
    @Resource int maxAcceptableFails;
    @Resource int minOfPositivePoints;

    public String executeEvaluation() {…}
}
```

Using Dependency Injection (Cont.)

```
<enterprise-beans>
 <session>
  <ejb-name>SalesProcessorBean</ejb-name>
  <ejb-class>ejb.samples.di.SalesProcessorBean</ejb-class>

  <env-entry>
        <description>
                The maximum of acceptable fails
        </description>
        <env-entry-name>maxAcceptableFails</env-entry-name>
        <env-entry-type>java.lang.Integer</env-entry-type>
        <env-entry-value>5</env-entry-value>
  </env-entry>
```

Using Interceptors

- AOP versus Interceptors
- Interceptor acting as
  - Business Validation
  - Audit
  - Security Issues
- The deployer can turn on or off the interceptors and define the execution order

Agenda

Brief Introduction to EJB 3 Technology New Features

EJB 3 Technology in the Real World

Session Beans Pitfalls, Tips, and Tricks

Entity Beans Pitfalls, Tips, and Tricks

Message Driven Beans Pitfalls, Tips, and Tricks

Refactoring to Better Use EJB 3 Technology

Old and New Design Patterns

Conclusion

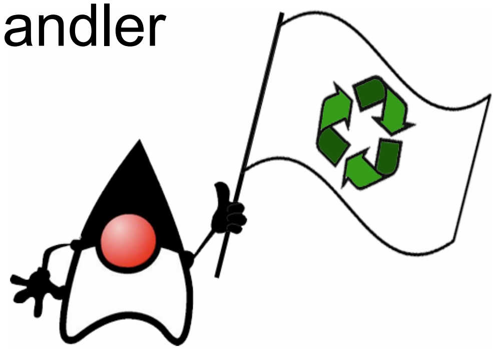
Old and New Design Patterns

Old design patterns applied to EJB 3.0 technology

- Session Façade
- Value Object
- Fast Lane Reader
- Business Delegate
- Value List Handler

New design patterns

- Business Rule Interceptor
- Entity View
- Eager Loading Parameter
- Data Change Observer
- Exportable Method
- Exportable Service Broker



Value Object/Data Transfer Object

- Usually you can pass your persistent POJOs to the next tier, but in some cases DTOs are still useful
  - Reduce the problems with lazy loading and detached objects, specially when the view and the persistence tiers are developed by different teams
  - When you have large entities (lots of properties, embedded objects, large text properties, etc.) and you need to decrease the amount of data transferred to another tier

Fast Lane Reader

- Goal: Data retrieval is done through a direct query to the database, without using EJB Entity Beans for this task
- In EJB 3 technology, this pattern is almost unnecessary, unless you want to use database proprietary features and get the maximum possible performance using Java DataBase Connectivity (JDBC™) directly

Business Delegate

- Goal: Reduce coupling between presentation tier and business services, hiding the underlying implementation details of the business service, such as lookup and access details of the EJB architecture
- It still useful to hide the technology used on the service layer from the presentation layer
- However, dependency injection makes the client side much easier, so, unless you really need the business delegate, try to avoid it

Business Rule Interceptor

- Problem 1: some business rules should be triggered depending on the current context
- Problem 2: new business rules should be added for a particular installation of your application

```
public void savePatientVisit(Visit v) {
    if(!currentUser().worksForHospital(v.getHospital) {
        throw new SecurityException(“can’t save data”);
    }
    if(v.getHospital().equals(“H1”)) {
         runHospital1BusinessRule(v);
    }
    save(visit);
}
```

Business Rule Interceptor (Co

- Solution: create an interceptor to handle these rules

```
@Interceptors({SecurityInterceptor.class,
               CustomRulesInterceptor.class})
public void savePatientVisit(Visit v) {
   save(visit);
}
```

- The interceptor can be integrated to a rules engine, such as Drools (JBoss Rules)

Entity View

- Problem: you have an entity bean with many properties, you want to retrieve only a few properties so you can send a smaller object to the next tier
- Solution: create a query that retrieves a smaller object

```
SELECT NEW br.com.zilics.PatientView(p.id, p.name)
FROM Patient p
WHERE p.name LIKE :name
```

Exportable Method

- Problem
  - You have to export a method to a Web Service but it has too many complex data types

```
public Proposal getProposalByCode(String code) {
   return em.find(Proposal, code);
}

@WebMethod
public String getProposalByCodeAsXML(String code) {
   return  XmlTransformer().marshal(getProposalByCode(code));
}
```

Exportable Service Broker

- Problem
  - You want to export your application to a web service, providing only one entry point
- Solution
  - Create a SLSB with simple signature structure methods and use Dependency Injection to access others EJB technology

Agenda

Brief Introduction to EJB 3 Technology New Features

EJB 3 Technology in the Real World

Session Beans Pitfalls, Tips, and Tricks

Entity Beans Pitfalls, Tips, and Tricks

Message Driven Beans Pitfalls, Tips, and Tricks

Refactoring to Better Use EJB 3 Technology

Old and New Design Patterns

Conclusion

Transition Impact for the Developers

Common mistakes of EJB 2.1 technology developers

- Avoiding inheritance
- Over-complex architectures
- Handling Lazy Loading

Summary

- EJB 3 Technology Benefits
  - Easier development model
  - Better object oriented development support
  - More productivity for developers
- EJB 3 Technology Drawbacks
  - Still missing an efficient standard pagination mechanism
  - Retrieving meta-information about the domain model in runtime is not standardized
  - Lazy loading handling still brings problems

For More Information

See

- Java Specification Request (JSR) 220 (http://jcp.org/en/jsr/detail?id=220)
- BOF-4834—Designing Self-Evolving and Self-Configuring Java Platform, Enterprise Edition (Java EE) Applications
- TS-4247—Enterprise JavaBeans 3.1 Technology
- Meet us at the java.net Community Corner!

Q&A

Fabiane Bizinella Nardon (fabiane@dev.java.net)

Edgar Silva (edgar.silva@redhat.com)

Implementing Java EE Applications Using Enterprise JavaBeans (EJB) 3 Technology: Real World Tips, Tricks, and New Design Patterns

Fabiane Bizinella Nardon

CTO
Zilics

www.zilics.com

Edgar Silva

Solutions Architect
JBoss (Red Hat)

www.jboss.com

Session TS-4721