Enterprise JavaBeans(tm) 3.0

Carlo de Wolf

Red Hat Middleware
EJB3 Lead Developer

February 13th 2008

Why Enterprise JavaBeans$^{(tm)}$?

- Allow rapid development of reusable business components
- Using an easy infrastructure which does:
  - Memory management
  - Remote invocation
  - Thread management
- Thus having a predictable QoS
- Without any low-level system code

What is an Enterprise JavaBeans$^{(tm)}$?

- It's not a POJO!
- It consists of a class + interceptor classes + interfaces
- Construction is different
- Invocation is different
- It's an assembly with one or more views

Life-cycle of an Enterprise JavaBeans(tm)

- No argument constructor
- Injection
- @PostConstruct life cycle interceptor(s)
- ~~@Init life cycle interceptor for stateful EJB~~
- “I'm alive!”

  Ready for method invocations
- @PreDestroy life cycle interceptor(s)

JBoss® WORLD ORLANDO 2008 PRESENTED BY RED HAT

Injection

- Applies to all JavaEE container managed objects
- @Resource
- @EJB
- Injection does not always happen!
- *@PostConstruct*
- *@PreDestroy*

Interceptors

- Type of interceptor
  - Life-cycle : intercept the construction or destruction of a bean
  - Business Method : intercept method invocations on a bean
- Level of interception : default, class, method (next page)
- Are only effective on “public” methods

Interceptor Levels

- System : interceptors defined in ejb3-interceptors-aop.xml
- Default : interceptors bound to all EJBs
- Class : interceptors bound to one bean(!)
- Method : interceptors bound to one method
- Default and class level interceptors can be excluded
- Only default, class and bean interceptors can be sorted

Intermezzo : EJB 3 == IoC?

- Dependency Injection
- Don't call use, we call you
- The Invocation difference : detached instances

Enterprise JavaBeans(tm) Types

- Stateless Session Bean (Shared)
- Stateful Session Bean (Unique)
- Message Driven Bean
- ~~Entity Bean~~
- Service Bean* / Singleton Bean
- Consumer Bean*

Session Bean

- Never without state
- Stateless : instance pooling
- Stateful : instance caching

Session Bean Client Views

- Business interface views
  - Local
  - Remote
- ~~EJB 2.1 views (local interface & remote interface)~~
- WebService view
- Management view (JMX)
- EJB 3.1: no-interface view

Message Driven Bean

- Asynchronous invocation via messages
- No direct invocation
- No client visible view

Service Bean / Singleton Bean

- One instance to serve all
- Service Bean is thread safe (thus a bottle neck)
- Singleton Bean has declarative concurrency

Consumer Bean

- Asynchronous invocation via interface
- Provides an asynchronous view

Bean Context

- Who is calling me?
- What is being called?
- Interact with timers
- Influence the transaction outcome
- The flavors:
  - EJBContext
  - SessionContext (extends EJBContext)
  - MessageDrivenContext (extends EJBContext)
  - ~~EntityContext (extends EJBContext)~~

EJBContext

- Applies to every enterprise bean type with exceptions
- getCallerIdentity: who called me?
- setRollbackOnly: mark the current transaction as not-committable
- getTimerService : get the timer service (except stateful)

SessionContext

- getInvokedBusinessInterface: what is called?
- Obtain a proxy to the current bean

Transactions

- Mandatory : if no active-tx, throw TxRequired
- <u>Required</u> : if no active-tx, begin new tx
- RequiresNew : suspend active-tx*, begin new tx
- Supports : do nothing (=> unspecified tx context!)
- NotSupported : suspend active-tx*
- Never : if active-tx, throw EJBException
- Transaction timeout requires a new transaction!

Persistence

- Using Java Persistence API
- Per default JTA transaction type
- Allows for Extended Persistence Context

Security

- Authentication is handled through JAAS
- @SecurityDomain to specify the JAAS application policy
- Declarative security through annotations based on roles
- Assume a different role with @RunAs
- Missing: imperative security

Clustering

- Applies only to Stateful Session Beans
- High availability
- High performance

Asynchronous

- Async Session bean invocation
- Consumer Bean
- EJB 3.1: Future<V> methods

Performance & Tuning

- Average pool size
- Average execution time
- Average waiting time

Current State

- JBoss Application Server 4.2 / 4.3

  => JBoss Enterprise Application Platform 4.2 / 4.3
- JBoss Application Server 5.0
- JBoss Embedded (was JBoss EJB 3 Embedded)
- JBoss EJB 3 Plugin
- JBoss EJB 3 Standalone

Future Features

- What might come in future releases?
- @ContainerInterceptors
- Meta data inspection
- Instance pooling per user / reserved slots

Contributing

- Discussion on the forum
- JIRA
- Patches
  - Git?
- Becoming a submitter

Questions?