

EJB 3 Development in JBoss Environment

By : Yanai Franchi, Senior Software Engineer
"Tikal"



Agenda



- ▶ Preface
- ▶ EJB3 Core
- ▶ Demo
- ▶ Advanced EJB3
- ▶ Summary





Preface

EJB 2.1 Ailments

- ▶ EJB 2.1 is 'noisy'
 - » EJB-Interfaces and Home-Interfaces (remote & Local)
 - » XML Hell: Deployment descriptors (standard and proprietary)
 - » Over verbose, complicated API
 - » Intrusive framework – Lacks separation of concerns
- ▶ Entity Beans:
 - » OO Killer
 - » Not portable
 - » Barely testable
 - » Incurs serious overhead



EJB 2.1 Ailments- Cont'

.....

- ▶ Many LoC to implement an EJB
- ▶ Many LoC to write an EJB Client
- ▶ Entity-Beans dictates usage of DTOs
 - » “Shotgun Smell”
 - » Anaemic value objects with no behaviour
- ▶ Long development cycle: edit-compile-deploy-test



EJB 3.0 Remedies

- ▶ Everything is a POJO
- ▶ Facilitates Test Driven Development
- ▶ Simplified programming model
 - » No EJB-Interface to implement or extend
 - » Homeless
 - » No EJB callback methods (ejbCreate(), ejbRemove()...)
 - » Uses Java 5 Annotations (optional deployment descriptors)
 - » Sensible default values
 - » Dependency Injection
 - » Runtime Exceptions



EJB3.0 with JBoss 4.x

- ▶ EJB3 container can be integrated in JBoss-4.0.5
- ▶ Hibernate 3.2 as persistent provider.
- ▶ Extra services extending EJB3.0 standards
 - » Clustering
 - » Asynchronous calls to Session Beans.
 - » Managed Service POJOs
 - » Message Driven POJO (MDP)



JBoss AS 5.0-beta1 is out

- ▶ JBoss Microcontainer –
 - » POJO based microcontainer removing the dependency on JMX
- ▶ EJB3 container integrated
- ▶ Hibernate 3.2 – JPA certified
- ▶ JBoss-Messaging 1.2 –
 - » a complete rewrite of JBossMQ.
- ▶ JBoss WebServices 2.0 –
 - » New custom built JAX-WS compliant WebServices stack.
- ▶ JBossCache 2.0
 - » Traditional tree-structured and PojoCache support





EJB3 Core

Stateless Session Bean

```
@Remote  
@Local  
public interface HotelService {  
    List<Hotel> findAllHotels();  
    void saveHotel(Hotel newHotel);  
}
```

```
@Stateless  
public class HotelServiceBean implements HotelService {  
    public List<Hotel> findAllHotels() {  
        //find all Hotels from the DB  
    }  
    public void saveHotel(Hotel newHotel) {  
        //save hotel  
    }  
}
```



The Client Code

- ▶ Beans are created without home
- ▶ You still have to create InitialContext, lookup (No Dependency Injection) and do downcasts ☹️

```
public class HotelServiceTest extends TestCase {  
    public void testSaveHotel() throws Exception {  
        InitialContext ctx = new InitialContext();  
        HotelService hotelService = (HotelService)  
            ctx.lookup("hotel-app/HotelServiceBean/remote");  
        List<Hotel> hotels = hotelService.findHotels();  
    }  
}
```

Jndi
Name



Stateful Session Bean

```
public interface Cart {  
    void addItem(int prodId, int quantity);  
    void checkout();  
}
```

```
@Stateful  
public class CartBean implements Cart {  
    private List<Item> items = new LinkedList<Item>();  
    public void addItem(int prodId, int quantity) {  
        ...  
    }  
    @Remove  
    public void checkout() {  
        ...  
    }  
}
```

Message Driven Bean

```
@MessageDriven( activationConfig = {
    @ActivationConfigProperty(
        propertyName="destinationType",
        propertyValue="javax.jms.Queue"),
    @ActivationConfigProperty(
        propertyName="destination",
        propertyValue="queue/booking")
})
public class HotelBookingProcessorBean
    implements MessageListener {
    void onMessage(Message msg) {
        // check user credit with bank system
        // send confirmation via mail
        // other business logic
    }
}
```



What is JPA ?

.....

- ▶ JPA is part of the EJB3 standard (JSR-220)
 - » The standard ORM API for Java EE
 - » Simplifies development of JEE/JSE applications via object persistence
- ▶ Originally, an CMP improvement
 - » Became a separate persistence spec for POJOs
- ▶ Usable both with JSE5 and JEE5
- ▶ Implementation products
 - » Hibernate (JBoss)
 - » TopLink JPA (Oracle)
 - » OpenJPA (Apache)

Persistence Beans

- ▶ Everything is POJO: No interfaces to implement
- ▶ Supports all relationships (with cascading)
- ▶ Inheritance + Polymorphic Queries
- ▶ Eager and Lazy fetching
- ▶ Annotation based (XML is optional)
- ▶ Simple, lifecycle model – New, Persistent, Detached
- ▶ Provides a modern ORM programming model



Persistence Annotations

```
@Entity @Table(name = "HOTELS")
public class Hotel implements Serializable {
    private Long id;
    private String name;
    private String address;
    @Id @GeneratedValue
    public Long getId() {
        return id;
    }
    private void setId(Long id) {
        this.id = id;
    }
    @Column(name="NAME", length=40)
    public String getName() {
        return name;
    }
    @OneToMany(cascade=ALL) @JoinColumn(name="ITEM_ID")
    public Set<Bid> getBids() {
        return bids;
    }
}
```

EntityManager API

- ▶ Take control the lifecycle entities
- ▶ Execute queries
- ▶ All access through this service
 - » Creation, retrieval, removal, and merging
- ▶ Analogous to Hibernate Session



Persistence Context

- ▶ What is Persistence Context?
 - » A set of managed entity instances
 - » Persistent identity \Leftrightarrow object identity
 - » Analogous to Hibernate Session cache
 - » The Entity Manager keep the persistent context.

- ▶ What is the scope of Persistence Context?
 - » Transaction scope?
 - » Other scope?
 - » In JPA, a persistence context may span multiple (non-concurrent) transactions



JPA in Action

```
@Stateless
public class HotelServiceBean implements HotelService {
    @PersistenceContext
    private EntityManager em;

    public List<Hotel> findAllHotels() {
        return em.createQuery(
            "select h from Hotel h order by h.nameasc")
            .getResultList();
    }

    public void saveHotel(Hotel newHotel) {
        em.persist(newHotel);
    }
    ...
}
```

What is Embeddable JBoss?

.....

- ▶ Java EE 5.0 application servers are no longer the monolithic beasts of the J2EE 1.4 era.
- ▶ Embeddable JBoss can run EJB3 in
 - » Unit test environment
 - » Web Server
 - » JSE5 application.
 - » Other Application Servers
- ▶ Built on top of the new JBoss Microcontainer.
- ▶ Currently enabled services are JNDI, JCA, JTA, and the EJB 3.0 container



Embeddable Current State

.....

- ▶ Local JNDI
- ▶ Transaction Manager
- ▶ Local JMS
- ▶ Local TX datasource/
connection pool
- ▶ Stateful, Stateless,
Service, Consumer,
Producer, and MDBs
- ▶ EJB 3 Persistence
- ▶ Hibernate integration
- ▶ EJB Security
- ▶ XA Connection pool is
not available yet.
- ▶ Distributed remote
communication is not
supported yet.
- ▶ JNDI is not available
remotely
- ▶ You cannot access JMS
remotely.
- ▶ Consider it an alpha
release...



Running Embeddable JBoss

.....

- ▶ EJB3StandaloneBootstrap class bootstraps the ejb3 container.
- ▶ It find EJB and resources by ClassLoader.getResource.

```
EJB3StandaloneBootstrap.boot(null);
EJB3StandaloneDeployer deployer = new EJB3StandaloneDeployer();
//Add configuration files
deployer.create();
deployer.start();
EJB3StandaloneBootstrap.scanClasspath();
```



Embeddable JBoss in Web Application

- ▶ JBoss comes with a context listener class that you can use to configure your application.
- ▶ The *ServletBootstrapListener* will automatically scan all Jars within *WEB-INF/lib* for EJBs and Entity beans that can be deployed.

```
<listener>
  <listener-class>
    org.jboss.ejb3.embedded.ServletBootstrapListener
  </listener-class>
</listener>

<context-param>
  <param-name>jboss-kernel-deployments</param-name>
  <param-value>
    embedded-jboss-beans.xml, jboss-jms-beans.xml
  </param-value>
</context-param>
```



Demo



Advanced EJB3

JBoss Asynchronous Calls

```
public class HotelServiceTest extends TestCase {
    public void testSaveHotel() throws Exception {
        InitialContext ctx = new InitialContext();
        HotelService hotelService = (HotelService)
            ctx.lookup("sample-ejb3/HotelServiceBean/remote");
        HotelService hotelAsynchService = (HotelService)
            Async.getAsynchronousProxy(hotelService);
        hotelAsynchService.findAllHotels(); //Asynch call

        Future future =
            ((AsyncProvider)hotelAsynchService).getFuture();

        //Do some other interesting stuff
        while (!future.isDone())
            Thread.sleep(100);
        List<Hotel> hotels = (List<Hotel>) future.get();
    }
}
```

JBoss Service POJOS

```
public interface HotelServiceManagement {  
    List<Hotel> findAllHotels();  
}
```

```
@Local (HotelService.class)  
@Service  
@Management (HotelServiceManagement)  
public class HotelServiceBean implements  
    HotelService, HotelServiceManagement {  
  
    public List<Hotel> findAllHotels() {  
        //find all Hotels from the DB  
    }  
  
    public void saveHotel(Hotel newHotel) {  
        //save hotel  
    }  
}
```



JBoss Message Driven POJOs (MDP)

```
@Producer public interface HotelBookingProcessor {  
    void processBooking(Booking booking) ;  
}
```

```
@Consumer ( activationConfig = {  
    @ActivationConfigProperty(  
        propertyName="destinationType",  
        propertyValue="javax.jms.Queue"),  
    @ActivationConfigProperty(  
        propertyName="destination",  
        propertyValue="queue/booking")  
})
```

```
public class HotelBookingProcessorPojo  
    implements HotelBookingProcessor {  
    public processBooking(Booking booking){  
        // check user credit with bank system  
        // send confirmation via mail  
    }  
}
```

JBoss MDP - Sender Side

.....

```
InitialContext ctx = new InitialContext();
HotelBookingProcessor hbp = (HotelBookingProcessor)
    ctx.lookup(HotelBookingProcessor.class.getName());

ProducerManager manager =
    ((ProducerObject)hbp).getProducerManager();

manager.connect();
try {
    hbp.processBooking(booking); //Asynch invocation!!!
} finally {
    manager.close(); // clean up the JMS connection
}
```



Transactions

- ▶ @TransactionalAttribute defines per-method transaction boundary
- ▶ Supported same propagation models as in 2.1:
REQUIRED, REQUIRES_NEW, SUPPORTS, MANDATORY, NEVER, NOT_SUPPORTED

```
@Stateless
public class HotelServiceBean implements HotelService {
    @TransactionalAttribute(
        TransactionAttributeType.REQUIRES_NEW)
    public void saveHotel(Hotel newHotel) {
        //save hotel
    }
}
```



Security

.....

- ▶ Specifies the list of roles permitted to access method(s) in an application
- ▶ Use `@RolesAllowed` for a list of security role names or `@PermitAll` to permit access for all.
- ▶ The `SecurityDomain` specifies the JAAS repository which will be used by JBoss to authenticate and authorize.

```
@org.jboss.ejb3.security.SecurityDomain("other")
@Stateless public class HotelServiceBean
    implements HotelService {
    @RolesAllowed( { "Role_Administrator" } )
    public void saveHotel(Hotel newHotel) {
        //save hotel
    }
}
```

Clustering

- ▶ For Session beans use the JBoss `@Clustered` annotation on the bean class.
- ▶ A clustered bean has load balancing and failover of a request.

```
@Remote (HotelService.class)
@org.jboss.annotation.ejb.Clustered
@Stateless public class HotelServiceBean implements HotelService {

    public List<Hotel> findAllHotels() {
        //find all Hotels from the DB
    }

    public void saveHotel(Hotel newHotel) {
        //save hotel
    }
}
```

Web-Services with EJB3

```
@WebService @SOAPBinding(style=Style.RPC)
public interface HotelService extends Remote {
    @WebMethod List<Hotel> findAllHotels();
    @WebMethod void saveHotel(Hotel newHotel);
}
```

```
@Stateless
@WebService(endpointInterface="com.tikal.sample.HotelService")
public class HotelServiceBean implements HotelService {
    public List<Hotel> findAllHotels() {
        //find all Hotels from the DB
    }
    public void saveHotel(Hotel newHotel) {
        //save hotel
    }
}
```



Dependency Injection

- ▶ Dependency Injection
 - » Bean class specifies dependencies instead of lookup
- ▶ On Jboss 5.x @EJB will work inside a servlet or JSP also.

```
@Stateful
public class HotelBookingProcessorBean
    implements HotelBookingProcessor {
    @EJB (name="CreditProcessorEJB")
    private CreditCardProcessor processor;

    @Resource (jndiName="java:/DefaultDS")
    private DataSource jdbc;

    ...
}
```



Callbacks

- ▶ Optional Annotations replace callback methods
 - » PostConstruct
 - » PreDestroy
 - » PostActivate
 - » PrePassivate

```
@Stateful
public class FacadeBean implements Facade {
    private URL propertiesURL;

    @PostConstruct
    public void init() {
        propertiesURL=getClass().getClassLoader().
            getResource("MyProperties.txt");
    }
}
```

Timer Callback

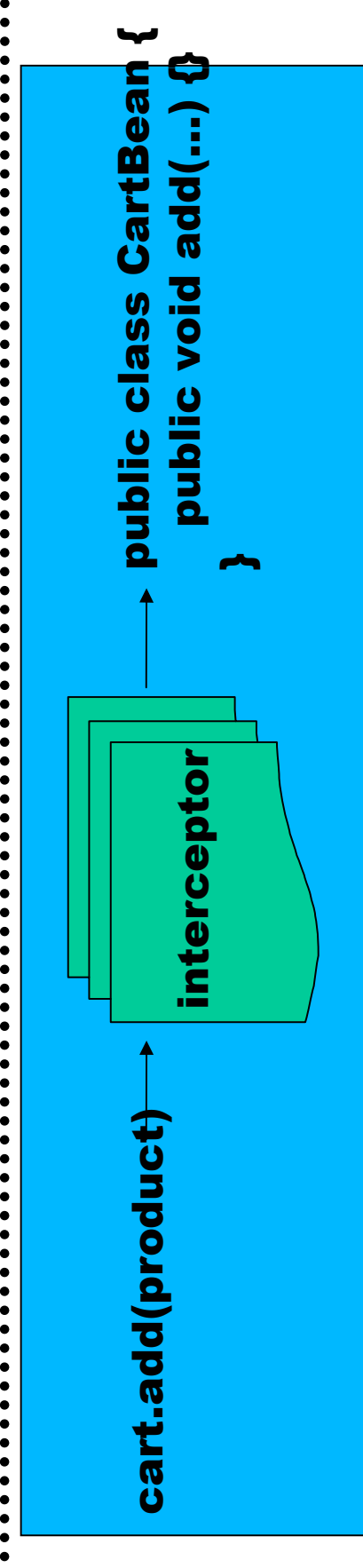
.....

```
@Stateless
public class ExampleTimerBean implements ExampleTimer {
    @Resource
    private SessionContext ctx;

    public void scheduleTimer(long milliseconds) {
        ctx.getTimerService().createTimer
            (new Date(new Date().getTime() +
                milliseconds), "Hello World");
    }

    @Timeout
    public void timeoutHandler(Timer timer) {
        System.out.println(timer.getInfo());
        timer.cancel();
    }
}
```

Interceptors



- ▶ Interceptors are executed in the gap between caller and bean
- ▶ Available only for session-beans and MDB
- ▶ Provide basic AOP
- ▶ Allow custom code to be applied to an EJB
- ▶ Simply add `@AroundInvoke` to a method
- ▶ A custom interceptor class can be defined



Interceptors

```
public class TracingInterceptor {  
    @AroundInvoke  
    public Object log(InvocationContext ctx){  
        long start = System.currentTimeMillis();  
        try {  
            return ctx.proceed();  
        } catch (Exception e) {  
            throw new RuntimeException(e);  
        } finally {  
            long time = System.currentTimeMillis() - start;  
            String method =  
                ctx.getBean().getClass().getName() + "." +  
                ctx.getMethod().getName() + "()";  
            System.out.println(method+" took "+time+"ms");  
        }  
    }  
}
```



Interceptors Cont.

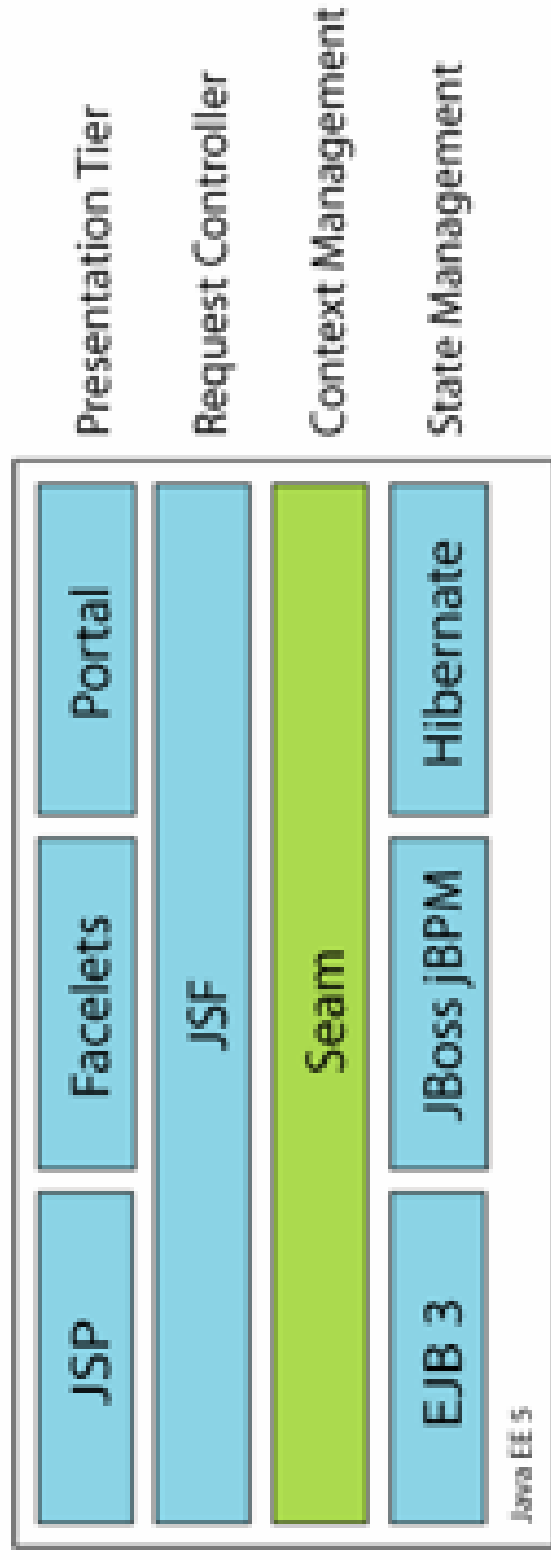
```
@Stateless
@Interceptors ({"com.tikal.sample.TracingInterceptor"})
public class HotelServiceBean implements HotelService {
    // ...
}
```

```
<assembly-descriptor>
...
<interceptor-binding>
  <ejb-name>
    con.tikal.sample.HotelServiceBean
  </ejb-name>
  <interceptor-class>
    con.tikal.sample.TracingInterceptor
  </interceptor-class>
</interceptor-binding>
...
</assembly-descriptor>
```



A Taste Of Seam

- ▶ JBoss's latest and greatest web framework
 - » Gavin King's new baby...
- ▶ Unifies EJB and Web component models
 - » Stateful EJBs can be JSF Backing Beans!
- ▶ Support the notion of *Contextual Components*
 - » May be bound to event, page, conversation, session, process or application Scopes



Summary – EJB3 Pros & Cons

.....

- ▶ Few implementations ▶ Standard!
- ▶ Recently released ▶ Stateful Model
- ▶ Small user base ▶ Can span multiple user requests.
- ▶ DI for JNDI only ▶ Enterprise Features
 - » Clustering
 - » JTA
 - » Built-in Security model
- ▶ Minimal AOP support
- ▶ ACL Security Standards
- ▶ Missing Service-
abstraction
- ▶ Too little, too late?





Q&A



Thank You

yanai@tikalk.com