Java EE6 and JBoss AS6 What's coming?

Jason T. Greene
JBoss, a Division of Red Hat

About The Speaker

- Lead of the JBoss Application Server project
- Member of JSR-316 EG (Java EE6 Spec)
- Member of JSR-299 EG (CDI [Web Beans])

http://in.relation.to/Bloggers/Jason

Primary Goals of EE6

- Extensibility
  - Allow more components to be standalone (EJB3.1)
- Profiles
  - Allow different subsets of JCP specifications
  - Web Profile being the first
- New Additions
  - Contextual Dependency Injection (Web Beans)
  - Bean Validation
  - JAX-RS

Notable Updates

- Servlet 3.0
  - JSR-250 - Common Annotations (Finally)
  - Async Support
- EJB 3.1
  - Singleton Component / Custom Concurrency
- JPA 2.0
  - Type-safe Criteria API
- JSF 2.0
  - AJAX Support

Web Profile Contents

- Data Persistence
  - JPA 2.0
  - JTA
- Component Framework
  - EJB Lite 3.1
  - Web Beans (TBD)
- Presentation
  - JSF 2
  - Servlet 3

EJB 3.1

- Embeddable / Standalone Usage
- New Singleton Session Bean
  - Custom Concurrency
- WAR based deployments
- EJB-Lite
- Asynchronous Methods
- Global/Portable JNDI names

Singleton Session Bean

```
@Startup @Singleton
public class SharedBean implements Shared {
  private int count;
  @PostConstruct void init() {
     count = 5;
  }

  @Lock(READ) public int getCount() {
     return count;
  }

  @Lock(WRITE) public int incrementCount() {
     return ++count;
  }
}
```

JPA 2.0

- Access Modes (field, property, etc)
- Orphan Removal
- Type-Safe Criteria
- Ordered Lists
- Locking API
- Cache API
- Integration with Bean Validation

Type-safe Query: Meta-model

- Generated from model using tooling

```
public class Item_ {
   public static Attribute<Item, Long> id;
   public static Attribute<Item, Boolean> shipped;
   public static Attribute<Item, String> name;
   public static Attribute<Item, BigDecimal> price;
   public static Map<Item, String, Object> photos;
   public static Attribute<Item, Order> order;
   public static Attribute<Item, Product> product;
}
```

Type-safe Query: Example

```
SELECT c.name
FROM Customer c JOIN c.orders o JOIN o.items i
WHERE i.product.productType = 'printer'
```

↓

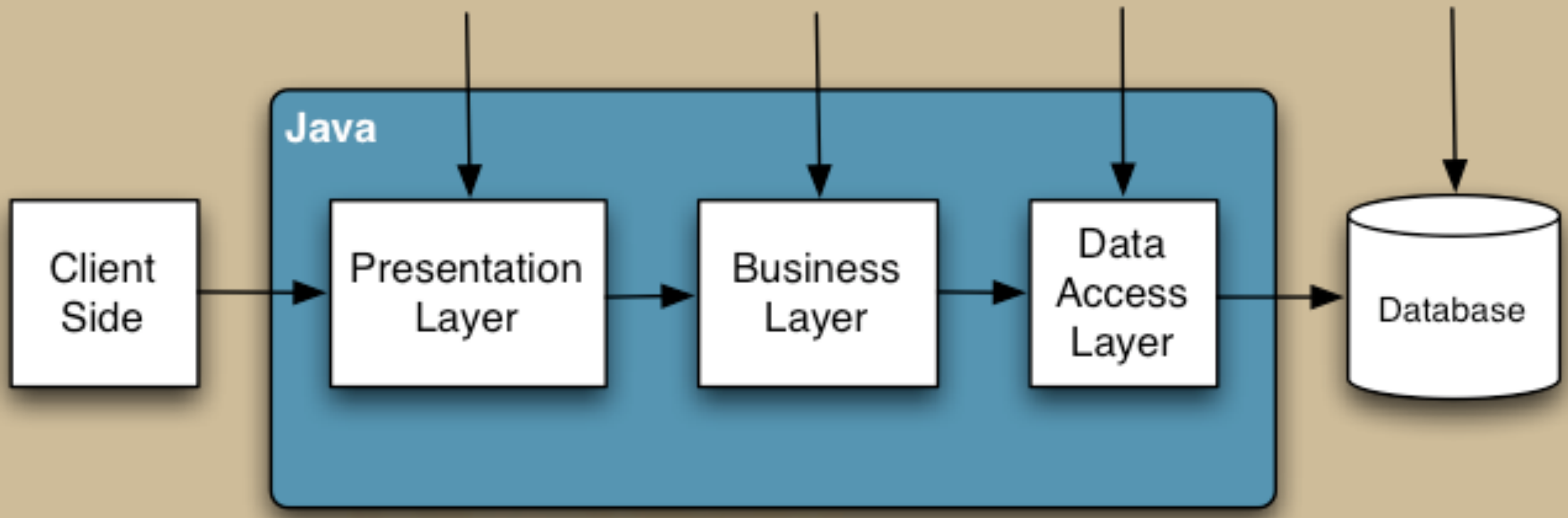
```
EntityManager em;
QueryBuilder qb = em.getQueryBuilder();

Query q = qb.create()
Root<Customer> cust = q.addRoot(Customer.class);
Path<Order, Item> item =
    cust.join(Customer_.orders).join(Order_.items);

q.select(cust.get(Customer_.name))
 .where(
    qb.equal(
        item.get(Item_.product).get(Product_.productType),
         "printer")
       );
```

Common Validation Points

- Each tier can/should be validated



- But, what happens when all use the same model?

Problems with current approach

- Duplication
  - multiple declarations of the same constraint
  - code duplication
  - risk of inconsistency
- Multiple runtime checking
  - not all constraints can be expressed by all engines
  - slightly different semantic?

JSR-303 Saves the day!

- Uniform way to express a constraint
  - everybody speaks the same language
  - based on the domain model (JavaBeans)
- Standard way to validate constraints
  - one runtime engine
  - same validation implementations shared
- Bridge for constraints out of Java land
  - API to access the constraint repository

Built-in Constraints

But you can write your own!

- In for sure
  - @NotNull / @Null
  - @Size
  - @AssertTrue / @AssertFalse
  - @Past / @Future
  - @Min / @Max
  - @Digits
- Maybe
  - @Pattern
  - @Like
  - @AlphaNumerical
  - @Email

Simple Example

```
public class Address {
  @NotNull
  @Size(max=30, message="longer than {max} characters")
 private String street1;
 private String street2;
  ...
}
```

JSR-299 JCDI (Web Beans)

- Injection for simple JavaBeans and EJBs
- Binding types
- Deployment types
- Scopes / Contexts
- Producers
- Interceptors & Decorators
- Stereotypes
- EL

Binding Types

```
@PayByCheque

public class ChequePaymentProcessor implements
PaymentProcessor {

    public void process(Payment payment) { ... }

}

@PayByCreditCard

public class CreditCardPaymentProcessor implements
PaymentProcessor {

    public void process(Payment payment) { ... }

}

@PayByCheque PaymentProcessor chequeProcessor;

@PayByCreditCard PaymentProcessor creditCardProcessor;
```

Deployment Types

```
@DeploymentType
public @interface Mock {}


@Mock
public class MockPaymentProcessor implements
PaymentProcessor {...}


<Deploy>
   <test:Mock/>
</Deploy>


@Current PaymentProcessor processor
```

Scopes

- Injectable bean instances are tied to a contextual lifespan, or ***scope***.
- Several pre-defined scopes
  - Request
  - Session
  - Conversation
  - Application

Scope Example

```
@RequestScoped
public class Credentials {...}

@SessionScoped
public class ShoppingCart {...}

@ConversationScoped
public class Order {...}

@ApplicationScoped
public class Catalog {...}
```

Producers

- Programatic control of instance creation

```
@Produces
public PaymentStrategy getPaymentStrategy() {
  swtich (paymentStrategy) {
    case CREDIT_CARD: return new CreditCardPaymentStrategy();
    case CHEQUE: return new ChequePaymentStrategy();
    case PAYPAL: return new PayPalPaymentStrategy();
    default: return null;
  }
}

@Produces
public PaymentProcessor getPaymentProcessor(
            @Synchronous PaymentProcessor sync,
           @Asynchronous PaymentProcessor async) {
    return isSynchronous() ? sync : async;
}
```

Decorators

- Like interceptors but type-safe

```
@Decorator
public abstract class LargeWithdrawDecorator
        implements Account {
    @Decorates Account account;
    public void withdraw(BigDecimal amount) {
        account.withdraw(amount);
        if ( amount.compareTo(LARGE_AMOUNT)>0 ) {
            Audit.alert(account, “Large Withdraw”);
        }
    }
}
```

Stereotypes

- Essentially meta-annotation macros

```
@Named
@RequestScoped
@Stereotype
@Target({TYPE, METHOD})
@Retention(RUNTIME)
public @interface Model {}
```

```
@Named
@RequestScoped                 ·····➤    @Model
public class JSFBean {}                   public class JSFBean {}
```

EL Integration

```
<h:dataTable value="#{cart.lineItems}" var="item">
    ....
</h:dataTable>


@SessionScoped @Named("cart")
public class ShoppingCart { ... }
```

JBoss AS6 Plans

- EE6 Compliance
- Improved OSGi support
- Embedded & "out of container" usage
- High-performance messaging rewrite (JBM2)
- Faster remoting implementation (R3)
- Very flexible profiles

Where to go from here

- JBoss.ORG (www.jboss.org)
- JSR-316 EE6 Spec
  - http://www.jcp.org/en/jsr/detail?id=316
- JSR-303 BV Spec
  - http://www.jcp.org/en/jsr/detail?id=303
- JSR-299 JCDI (Web Beans) Spec & RI (with docs)
  - http://www.jcp.org/en/jsr/detail?id=299
  - http://in.relation.to/Bloggers/FirstBetaOfWebBeansAvailable

Questions?

Q & A