JBoss WORLD
CHICAGO 2009

# FOLLOW US:

TWITTER.COM/REDHATSUMMIT

# TWEET ABOUT US:

ADD #SUMMIT AND/OR #JBOSSWORLD TO THE END OF YOUR EVENT-RELATED TWEET

presented by

JBoss WORLD
CHICAGO 2009

# Java EE6 & JBoss AS 6

## What's coming your way.

Jason T. Greene
JBoss, a Division of Red Hat

presented by

# About the Speaker

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

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

# JBoss AS vs JBoss EAP

- AS is the community driven application server
  - Focused on delivering the latest cutting-edge technology
  - Updates occur more frequently, and are feature-focused
  - Incubates new experimental approaches/technologies
- EAP is the hardened production fork of AS
  - Pulls in the stable pieces of AS
  - Provides steady/frequent bug-fix stream
  - Supported for many years
  - Backwards compatibility guarantees
  - Longer, more conservative release cycle

# JBoss AS vs JBossEAP (Cont).

- This presentation is focused on AS6 & EE6, not EAP
- However, EAP6 will be based on AS6

# 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
- Pruning
  - Define a removal process for defunct platform specs
  - CMP, JAX-RPC, JAXR, JSR-88 are “pruned” in EE6
- Technology Improvements
  - New Additions & Updates

# New Additions

- Managed Beans
- Contexts and 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 3.1 (Lite)
  - CDI (Web Beans)
- Presentation
  - JSF 2.0
  - Servlet 3.0

# Managed Beans

- EE POJOs with minimal container requirements
- Used to unify the many “simple” / “lightweight” objects introduced by various specs
  - JSF, 299, JAX-RS, JSR-181, etc
- Supports JSR-250 annotations (@PostConstruct)
- Optional naming @ManagedBean(“cart”)
- Supports interceptors
- EE Resource injection
- JCDI Injection

# EJB 3.1

- Embeddable / Standalone Usage
- New Singleton Session Bean
  - Custom Concurrency
- WAR based deployments
- EJB-Lite
- Asynchronous Methods
- Access Timeouts
- 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;
  }

  @AccessTimeout(1000)
  @Lock(WRITE)
  public int incrementCount() {
     return ++count;
  }
}
```

# JSR-299 Contexts and Dependency Injection

- Injection for simple JavaBeans and EJBs (and more)
- Binding types
- Alternatives
- Scopes / Contexts
- Producers
- Interceptors & Decorators
- Stereotypes
- EL
- More detail in WB Presentation on Thursday!

# 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

```
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

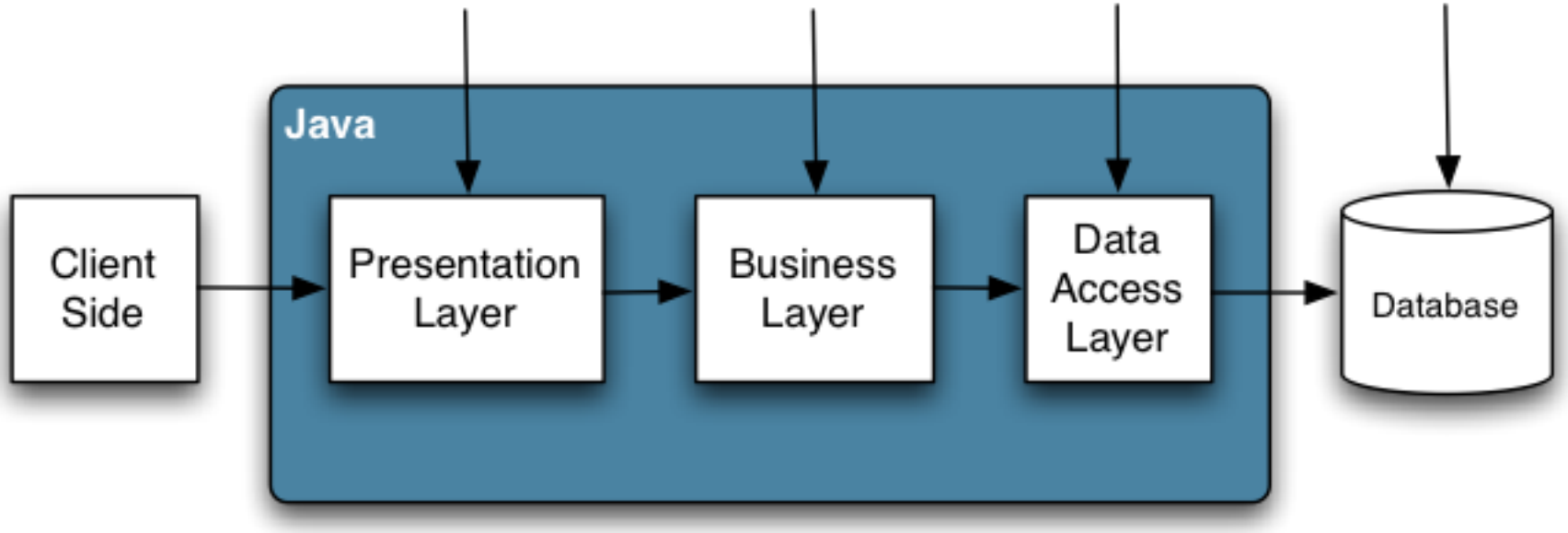
```
SELECT c.nameFROM Customer c JOIN c.orders o JOIN o.items
iWHERE i.product.productType = 'printer'
```

```
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 if each tier uses the same model object?

# Obvious Problems With Current Approach

- Duplication
  - Redundant Code
  - Each constraint expressed differently  (annoying)
  - Often inconsistent (bad)
  - Sometimes contradicting (very bad!)
- Multiple Validation Runtimes
  - Each engine evaluates differently
  - Each with their own limitations

# Bean Validation (JSR-303) saves the day!

- Uniform constraint expression
  - All layers understand the same validation “language”
  - Constraints in one place, the JavaBean model
  - Also expressible in XML
- Standardized constraint evaluation
  - One engine, so only one behavior
  - Engine is shared between all frameworks
- Standard validation APIs
  - Repository API allows querying constraints
  - Validation API validates constraints

# Built-in Constraints

- @NotNull / @Null
- @Size
- @AssertTrue / @AssertFalse
- @Past / @Future
- @Min / @Max
- @Digits
- @Pattern

# Also Supports Custom Constraints

```
/**  * Mark a String as representing a valid
legal
 * zip code
 */  @Constraint(validatedBy =
            ZipCodeValidator.class)
@Target({METHOD, FIELD, ANNOTATION_TYPE})
@Retention(RUNTIME) public @interface ZipCode {
     String countryCode();      String
message() default
            "{constraint.zipCode}";
Class<?>[] groups() default {}; }
 {}; }
```

# Simple Usage Example

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

 @ZipCode(countryCode="US")
 private String zip;

 ...
}
```

# AS6 Plans

- Centralized configuration
- Management Improvements
- Embedded & Unit testing support
- Performance improvements
- EE6 support
- OSGi support
- Separation of APIs and Internals
- Numerous component updates

# Centralized Configuration

- Domain based
  - One file for a whole cluster!
  - Hierarchal overrides (e.g. server specific setings)
  - Modifiable via embedded console, file, and CLI
- User friendly
  - Pure configuration (no server impl details)
  - Schema driven (tab completion)
- Controls shared / common resources
  - Ports
  - Threads
  - Logging

# Example configuration style

```
<threads xmlns="urn:jboss:threads:1.0">
   <thread-group name="SystemThreadGroup" group-name="System Threads"/>
   <thread-factory name="SystemThreadFactory" group="SystemThreadGroup"/>
   <thread-pool-executor name="ThreadPool" thread-factory="SystemThreadFactory"
                                 queue-length="1000">
        <core-pool-size count="5" per-cpu="2"/>
        <max-pool-size count="10" per-cpu="3"/>
        <keepalive-time time="30" unit="seconds"/>
        <reject-policy name="block"/>
    </thread-pool-executor>
</threads>
<xnio xmlns="urn:jboss:xnio:1.0">
   <tcp-server name="HttpConnectorTcpServer" backlog="500" reuse-address="true">
         <handler-factory-bean name="HttpConnectorHandler"/>
         <bind-address address="${jboss.server.bind.address}" port="8080"/>
   </tcp-server>
   <tcp-server name="RemotingConnectorTcpServer" backlog="100" reuse-address="true">
         <handler-factory-bean name="RemotingConnectorHandler"/>
         <bind-address address="${jboss.server.bind.address}" port="4446"/>
   </tcp-server>
</xnio>
```

# Embedded & Unit Testing

- Full (or partial) AS environment on Java SE
  - Can reuse existing classpath/clasloaders
  - All capabilities in AS can be used (not just servlet)
- Simple DSL style API
  - All AS configuration will be possible via API
  - Full control of server functions
    - startup/shutdown/deploy
  - Virtual deployment assembly
- Maven integration
  - Automated WAR testing, etc

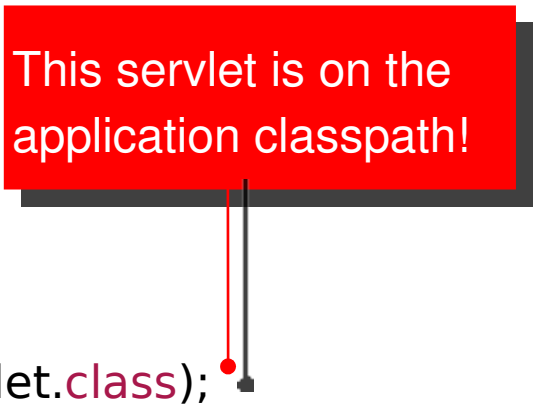
# Current Embedded Prototype

```
// Initialize
final ClassLoader cl = Thread.currentThread().getContextClassLoader();
final MCServer server = MCServerFactory.createServer(cl);

// Configure some stuff
server.getConfiguration().bootstrapHome("/path/")
  .bootstrapName("boootstrap.xml");
// Start the server
server.start();

// Build Archive
final Archive archive =
    VfsMemoryArchiveFactory.createVirtualArchive(name)
      .addResource(locationWebXml, newPathWebXml)
      .addResource(locationJsp, PATH_JSP).addClass(MyServlet.class);

// Deploy itserver.deploy(archive);
```



# Performance Improvements

- Work under-way on improving boot time
  - On-demand services & Lazy dependencies
  - Parallel startup
  - Flexible profiles
  - Annotation Indexing
- Remoting 3 & JBoss Marshalling
  - Much faster remote invocation
  - Special marshalling API for fastest possible serialization
  - Also JDK serialization compatible, but much faster

# Performance Improvements (Cont.)

- JBoss Messaging 2
  - High performance rewrite
  - Provides native plugins for optimal kernel hooks
  - Very fast tx log impl using Direct I/O
- Faster Clustering
  - Based on Infinispan technology
  - Extreme scalability (way beyond buddy replication)
  - Smaller memory footprint

# Separation of APIs and Internals

- All AS services will switch to the Module based classloader
  - Only JBoss API classes will be visible to applications
  - Applications can go beyond the EE rules and do the same
- Separate locations for internal JBoss artrifacts, system configuration, and application deployments
- Centralized Configuration will be an API
- Cleaner dependencies
- JBoss Tattletail tool

# AS 5.2

- Major EE6 features
  - CDI / Web Beans
  - Bean Validation
  - JSF 2
  - JPA 2
- Mod_cluster
- Embedded Prototype
- Embedded console improvements
- Logging Improvements

# Where to go from here.

- Check out the EE6 spec drafts at www.jcp.org
- Check out the community www.jboss.org
- User forums www.jboss.org/forums
- #jboss on irc.freenode.org
- Want to contribute?
  - #jboss-dev on irc.freenode.org
  - jboss-development ml at lists.jboss.org
  - Developer forums www.jboss.org/forums

# QUESTIONS?

## TELL US WHAT YOU THINK:
## REDHAT.COM/JBOSSWORLD-SURVEY