Apache OpenJPA

http://openjpa.apache.org/

Bean Validation Integration in JPA 2.0

July 17, 2009

Copyright © 2009, The Apache Software Foundation

Legal

- This presentation is based on Early Access levels of the following specifications:
  - JSR 317: Java Persistence API 2.0 – PFD (20090313)
  - JSR 303: Bean Validation – PFD (1.0.CR1 20090316)
  - Which require the following notice:
    - This is an implementation of an early-draft specification developed under the Java Community Process (JCP) and is made available for testing and evaluation purposes only. The code is not compatible with any specification of the JCP.
- Presentation is released under the Apache Software License 2.0
  - Copyright © 2009, The Apache Software Foundation
  - Apache and the Apache feather logo are trademarks of Apache Software Foundation.
  - Authored by: Jeremy Bauer and Donald Woods

Copyright © 2009, The Apache Software Foundation

Agenda

1 - Overview of JSR-303 Bean Validation 1.0

2 - JSR-317 JPA 2.0 support for Bean Validation

3 – OpenJPA modifications for Bean Validation

4 - Concerns and Issues

Copyright © 2009, The Apache Software Foundation

1 - Overview of JSR-303 Bean Validation 1.0

- 1.1 - Bean Validation Spec
- 1.2 – Constraint Definitions
- 1.3 – Constraint Descriptors
- 1.4 - Spec Required Constraints
- 1.5 - Validation Groups
- 1.6 - Spec Defined Exceptions

Copyright © 2009, The Apache Software Foundation

1.1 – Bean Validation Spec

- JSR-303 – JCP lead is Red Hat
- Hibernate Validation 4.0 will be the RI
  - The PFD/1.0 CR1 was the last publicized version from JCP website.
  - Current validation-api source is hosted in a public repo, so spec updates since the PFD can be tracked.
- Agimatec-Validation by agimatec GmbH on Google Code is the only other implementation we've found.
  - Has not finished implementing all of the PFD items.
  - Does not use the TraversableResolver.

Copyright © 2009, The Apache Software Foundation

1.2 – Constraint Definitions

- A null element is considered to be valid.
- Each constraint supports a List<T>
- Implementations must provide a MessageResourceBundle with some common predefined constraint messages.
- Constraints can be declared on interfaces and are cumulative
  - Traversable fields, traversable methods (getters only), classes (interfaces and superclasses) and traversable associations
  - Uses TraversableResolver.isTraversable() to determine if a given property should be accessed.
- Constraints are not processed in any particular order

Copyright © 2009, The Apache Software Foundation

1.3 – Constraint Descriptors

- Constraints can be provided by annotation or XML
  - No Spec requirement for Java SE 6
  - META-INF/validation.xml
  - XML overrides annotations unless ignore-annotations is set to false on the class descriptors
  - Descriptors can only be provided for a given class once
- Invalid arguments lead to a IllegalArgumentException, ConstraintDeclarationException or ValidationException
- A property can have constraints on both fields and methods

Copyright © 2009, The Apache Software Foundation

1.4 – Spec Required Constraints

- **@AssertTrue/AssertFalse(Boolean value)** - Boolean
- **@DecimalMax/DecimalMin(String value)** – BigDecimal, BigInteger, String, byte/Byte, short/Short, int/Integer, long/Long
- **@Digits(int integer, int fraction)** - BigDecimal, BigInteger, String, byte/Byte, short/Short, int/Integer, long/Long
- **@Future/Past()** – Date, Calendar
- **@Max/Min(long value)** – BigDecimal, BigInteger, String, byte/Byte, short/Short, int/Integer, long/Long
- **@Null/NotNull()** – Object
- **@Pattern(String regexp, Flag flags)** - String
- **@Size(int min, int max)** – String, Collection, Map, Array.length

Copyright © 2009, The Apache Software Foundation

1.5 – Validation Groups

- Default group includes all constraints
- Uses interfaces to define subsets of constraints
- Can inherit from other groups
- GroupSequence can be used to redefine the Default group for a class
- GroupSequence controls the order groups are processed and is the only way to define constraint ordering (one constraint per group)

Copyright © 2009, The Apache Software Foundation

1.6 – Validation Exceptions

- Runtime exceptions
  - ConstraintViolationException – generated by the framework (JPA2) if validation failures occur and contains the set of specific ConstraintViolation(s)
  - ConstraintViolation – contains the failure details: constraint descriptor, message, class, property and value
- Compile time exceptions (annotation processor)
  - ConstraintDefinitionException – illegal constraint
  - ConstraintDeclarationException – invalid constraint argument
  - UnexpectedTypeException – invalid property type
  - GroupDefinitionException – cyclic graph, illegal override

Copyright © 2009, The Apache Software Foundation

2 - JSR-317 JPA 2.0 support for Bean Validation

- 2.1 – Validation Overview
- 2.2 – Integration Diagram
- 2.3 – Validator Factory
- 2.4 – Validation Modes
- 2.5 – Validation Groups
- 2.6 – Validation Exceptions

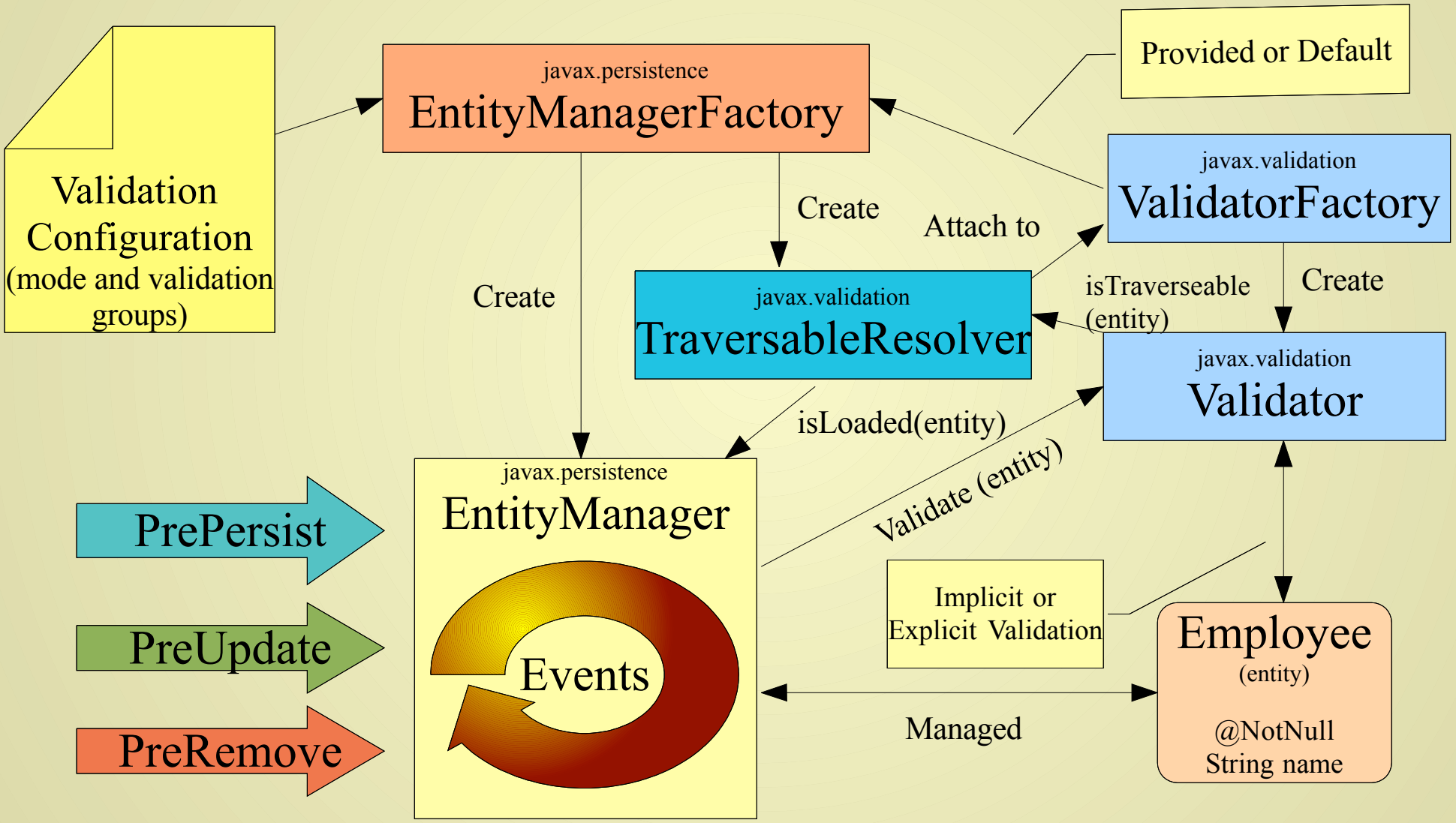
Copyright © 2009, The Apache Software Foundation

2.1 – Validation Overview

- **Validation is optional.** The JPA 2.0 Spec does not require a Bean Validation implementation.
- A TraversableResolver must be supplied by the persistence provider, so validation:
  - Does not cause unloaded attributes to be loaded (confirm to FetchType.Lazy/Eager)
  - Validation cascading or embedded attributes (either marked with @Valid) does not traverse entity associations

Copyright © 2009, The Apache Software Foundation

2.2 – Integration Diagram



Copyright © 2009, The Apache Software Foundation

2.3 – Validator Factory

- Java EE containers and Java SE applications can provide a javax.persistence.validation.factory in the EMF properties Map.
- A default instance is obtained from the Validation implementation in the classloader (if one is present) if none are supplied.

Copyright © 2009, The Apache Software Foundation

2.4 - Validation Modes

- javax.persistence.ValidationMode
  - **Auto** (default) – if a validation provider is available, then validation should occur
  - **Callback** – validation is required and a PersistenceException must be thrown if a provider cannot be obtained
  - **None** – no validation should be attempted and the lack of a validation provider should not cause an exception
- Can be set per PU
  - <validation-mode> element in the persistence.xml
  - javax.persistence.validation.mode in the EMF properties Map
- EMF supplied properties will override the XML

Copyright © 2009, The Apache Software Foundation

2.5 - Validation Groups

- Defines validation groups for entity life-cycle events
  - javax.persistence.validation.group.**pre-persist** – Default validation group called after all other PrePersist callbacks.
  - javax.persistence.validation.group.**pre-update** – Default validation group called after all other PreUpdate callbacks.
  - javax.persistence.validation.group.**pre-remove** – Default validation group is **NOT** called after all other PreRemove callbacks.

Copyright © 2009, The Apache Software Foundation

2.6 – Validation Exceptions

- javax.persistence.**PersistenceException** – thrown if validation mode is Callback and a provider could not be obtained
- javax.validation.**ConstraintViolationException** – thrown if any constraint failures occur and contains the set of javax.validation.ConstraintViolation instance(s)

Copyright © 2009, The Apache Software Foundation

3 - OpenJPA modifications for Bean Validation

- 3.1 – Configuration Updates
- 3.2 – Integration Diagram
- 3.3 – LifecycleEventManager
- 3.4 – TraversableResolver
- 3.5 – Unit Testing

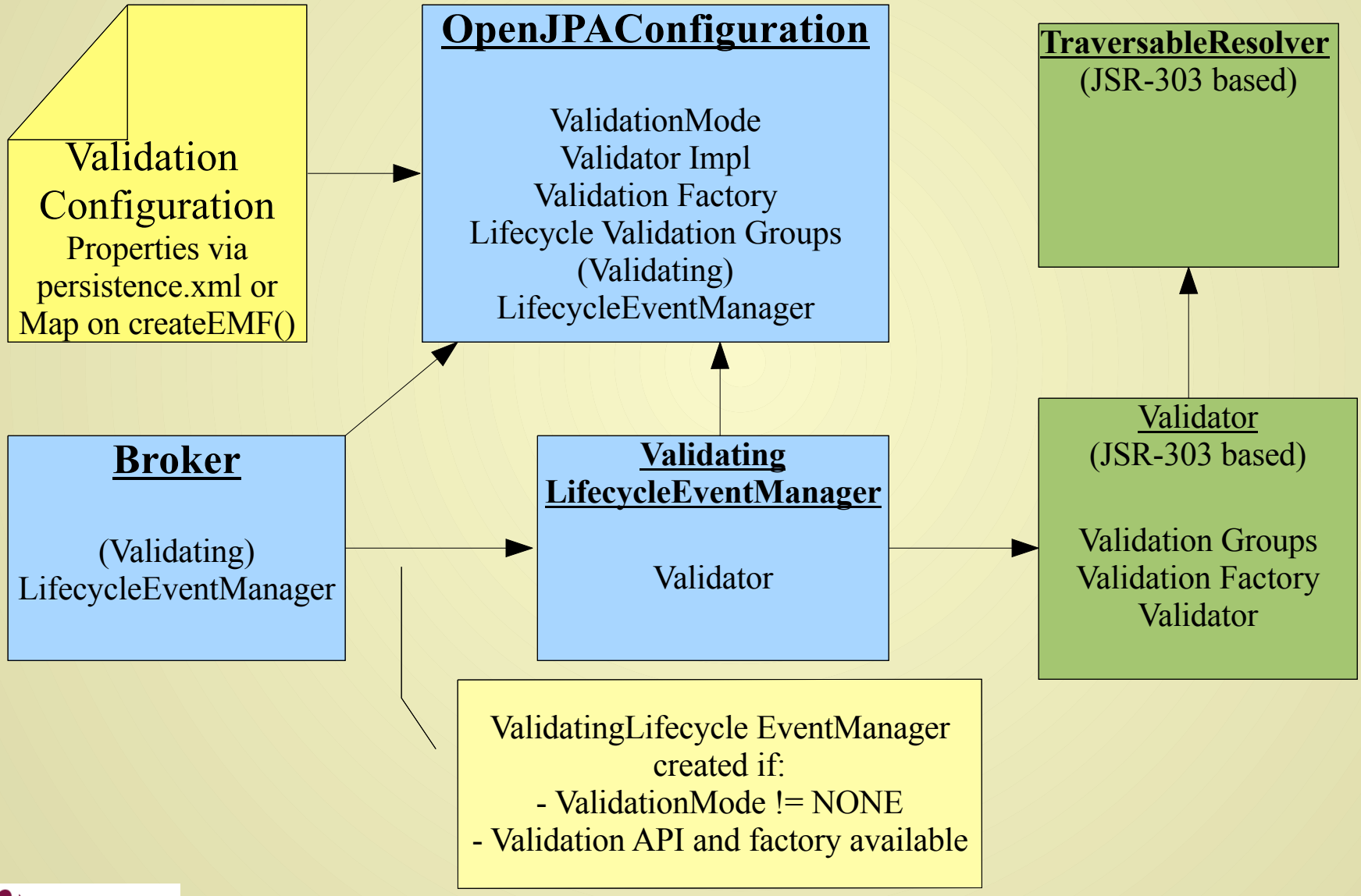
Copyright © 2009, The Apache Software Foundation

3.1 – Configuration Updates

- New configuration properties
  - javax.persistence.validation.factory – EMF property
  - javax.persistence.validation.mode – PU or EMF property
  - javax.persistence.validation.group.pre-persist – PU or EMF property and entity annotation
  - javax.persistence.validation.group.pre-update – PU or EMF property and entity annotation
  - javax.persistence.validation.group.pre-delete – PU or EMF property and entity annotation
  - openjpa.Validator - OpenJPAConfiguration()
  - openjpa.LifecycleEventManager - OpenJPAConfiguration()
- Pluggable Validator
  - Removes dependency on JSR-303 APIs
- Pluggable LifecycleEventManager
  - Loaded during EMF creation in PersistenceProviderImpl by loadValidator() just like loadAgent(), as old invocation location in BrokerImpl was restricted to kernel classes
  - Requires access to openjpa-persistence classes after all config derivations are loaded

Copyright © 2009, The Apache Software Foundation

3.2 – Integration Diagram



Copyright © 2009, The Apache Software Foundation

3.3 – LifecycleEventManager

- Extended to provide event based Validation
  - ValidatingLifecycleEventManager
- Validation mode and provider availability determine whether to use standard or validating event manager
  - Reflection used to determine existence of JSR-303 provider and API (through ValidationUtils)
  - Eliminates runtime dependency on API and provider
- Calls Validator upon lifecycle events
- Interacts with validation provider agnostic interface
  - Allows plugging in any validation implementation which implements OpenJPA's validation interface

Copyright © 2009, The Apache Software Foundation

3.4 – TraversableResolver

- Provided to ValidationFactory upon Validator creation
- Simple interface with single *isTraversable* method
- Primary role is to prevent loading of unloaded entities/ attributes and traversal to related entities
- Spec dictates the need for a provider specific resolver to meet loading and relationship traversal reqs.
  - OpenJPA will provide and register a TraversableResolver upon Validator creation
- Container vs. provider level requirements unclear

Copyright © 2009, The Apache Software Foundation

3.5 – Unit Testing

- openjpa-persistence-jdbc
  - Tests do not require a validation provider, but need the geronimo-validation spec
  - Basic ValidationMode tests
  - Exception tests for mode=callback but no provider
- openjpa-integration/validation
  - New integration module created to test with one or more validation providers (agimatec-validation or RI)
  - Tests spec defined constraints, validation groups and validator factory usage, along with any expected exceptions

Copyright © 2009, The Apache Software Foundation

The End

Copyright © 2009, The Apache Software Foundation