

ORACLE®



JavaOne™

ORACLE®

What's new in Java Persistence API (JSR 338)

Lukas Jungmann
EclipseLink JPA team lead
Oracle Czech
October 27, 2015

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

- 1 Overview
- 2 Status
- 3 What's next
- 4 Q&A

Overview

Overview

- Industry standard object/relational mapping for Java
- Delivers portability across providers and databases
- Increases developer productivity
- Available for use in both Java SE and Java EE environment

Overview

JPA 1.0

- 2003-2006 under JSR 220 - Enterprise JavaBeans 3.0
- TopLink Essentials as a reference implementation
- The Java Persistence API
- The query language - JPQL
- Object/relational mapping metadata

Overview

JPA 2.0

- 2007-2009 under JSR 317 - Java Persistence 2.0
- EclipseLink as a reference implementation
- Java Persistence Criteria API

Overview

JPA 2.0 - Enhancements

- Mappings
 - join table, unidirectional one-to-many with no join table, element collections, orphan removal, persistently ordered lists, derived identifiers
- Validation
 - allow additional lifecycle validation support
- Pessimistic locking
- Entity metamodel API

Overview

JPA 2.0 - Enhancements

- Shared Cache API
 - basic cache operations
 - extensible by vendors
- JPQL enhancements
 - date, time, timestamp literals, type, nullif, coalesce, case statement, map support, collection input parameters, index in a list, variables in the select clause

Overview

JPA 2.0 - Vendors

- BatooJPA
- DataNucleus
- EclipseLink
- Hibernate
- ObjectDB
- OpenJPA

Overview

JPA 2.1

- 2011-2013 under JSR 338 - Java Persistence 2.1
- Converters
 - allow custom code conversion between database and object types
- Schema generation
 - automatic table, index and schema generation
- Stored procedures
- Unsynchronized persistence context

Overview

JPA 2.1

- Entity Graphs
 - allow partial or specified fetching or merging of objects
- Constructor result mapping for SQLResultSetMapping
- Criteria Update/Delete
- JPQL/Criteria enhancements
 - arithmetic subqueries, generic database functions, join ON clause, TREAT option

Status

Java Persistence 2.1.NEXT

Overview

- 2.1.NEXT = 2.2 Maintenance release (MR)
- Small update to 2.1
- Main topics:
 - Java SE 8
 - Java EE 8
 - Improve integration with existing standards
 - Address feedback from the community

Java Persistence 2.2 MR

Overview

- CDI support in attribute converters
- Meta annotations
- Repeating annotations

Java Persistence 2.2 MR

CDI support in attribute converters

```
public class String2ByteConverter implements AttributeConverter<String, byte[]> {
    private CryptoBean bean = CDI.current().select(CryptoBean.class).get();

    @Override
    public byte[] convertToDatabaseColumn(String attribute) {
        return bean.encrypt(attribute);
    }

    @Override
    public String convertToEntityAttribute(byte[] dbData) {
        return bean.decrypt(dbData);
    }
}
```

Java Persistence 2.2 MR

CDI support in attribute converters

- Allow sharing conversion logic through multiple components

```
public class String2ByteConverter implements AttributeConverter<String, byte[]> {  
    @Inject private CryptoBean bean;  
  
    @Override  
    public byte[] convertToDatabaseColumn(String attribute) {  
        return bean.encrypt(attribute);  
    }  
  
    @Override  
    public String convertToEntityAttribute(byte[] dbData) {  
        return bean.decrypt(dbData);  
    }  
}
```

Java Persistence 2.2 MR

Meta annotations

```
public class AuditingListener {
    @Inject private LoggingSystem logger;

    @PrePersist
    public void prePersist(Object entity) {
        logger.log("about to persist an entity:"
            + entity.toString());
    }
}
```

```
@Entity
@EntityListeners(AuditingListener.class)
public class Account implements Serializable
{...}
```

```
@Entity
@EntityListeners(AuditingListener.class)
public class Person implements Serializable
{...}
```

```
@Entity
@EntityListeners(AuditingListener.class)
public class Car implements Serializable {...}
```

Java Persistence 2.2 MR

Meta annotations

- Allow annotation inheritance, abstraction and encapsulation

```
@Target(TYPE) @Retention(RUNTIME)  
@Entity  
@EntityListeners(AuditingListener.class)  
@interface @AuditedEntity {}
```

Java Persistence 2.2 MR

Meta annotations

```
public class AuditingListener {  
    @Inject private LoggingSystem logger;  
  
    @PrePersist  
    public void prePersist(Object entity) {  
        logger.log("about to persist an entity:"  
            + entity.toString());  
    }  
}
```

Java Persistence 2.2 MR

Meta annotations

```
public class AuditingListener {  
    @Inject private LoggingSystem logger;  
  
    @PrePersist  
    public void prePersist(Object entity) {  
        logger.log("about to persist an entity:"  
            + entity.toString());  
    }  
}
```

```
@Target(TYPE) @Retention(RUNTIME)  
@Entity  
@EntityListeners(AuditingListener.class)  
@interface @AuditedEntity {}
```

Java Persistence 2.2 MR

Meta annotations

```
public class AuditingListener {
    @Inject private LoggingSystem logger;

    @PrePersist
    public void prePersist(Object entity) {
        logger.log("about to persist an entity:"
            + entity.toString());
    }
}
```

```
@Target(TYPE) @Retention(RUNTIME)
@Entity
@EntityListeners(AuditingListener.class)
@interface @AuditedEntity {}
```

```
@AuditedEntity
public class Account implements
Serializable {...}
```

```
@AuditedEntity
public class Person implements Serializable
{...}
```

```
@AuditedEntity
public class Car implements Serializable
{...}
```


Java Persistence 2.2 MR

Meta annotations

```
@NamedEntityGraphs({
    @NamedEntityGraph(name = "actors",
        attributeNodes = {
            @NamedAttributeNode("actors")}),
    @NamedEntityGraph(name="actorsAndAwards",
        attributeNodes = {
            @NamedAttributeNode(value = "actors",
                subgraph = "actorsGraph")}),
    subgraphs = {
        @NamedSubgraph(name = "actorsGraph",
            attributeNodes = {
                @NamedAttributeNode("actorAwards")})
    }
})
```

```
@NamedEntityGraph(
    attributes = {
        "id", "name", "description",
        "subgraph.property",
        "anotherSubgraph.property"}
)
```

Java Persistence 2.2 MR

Repeating annotations

- Remove the requirement for container annotations

```
@Entity
@NamedQueries({
@NamedQuery(name="findByName", query="..."),
@NamedQuery(name="findById", query="...")})
public class Employee {...}
```

Java Persistence 2.2 MR

Repeating annotations

- Remove the requirement for container annotations

```
@Entity
@NamedQueries({
    @NamedQuery(name="findByName", query="..."),
    @NamedQuery(name="findById", query="...")})
public class Employee {...}
```

```
@Entity
@NamedQuery(name="findByName", query="...")
@NamedQuery(name="findById", query="...")
public class Employee {...}
```

What's next

Java Persistence 2.2 MR

Overview

- Date/Time support
- Stream support in Queries
- JPQL/SQL string from Query

Java Persistence 2.2 MR

Date/time support

```
@Entity
public class Bus {
    @Id long id;
    LocalDateTime departure;
}
```

```
@Converter(autoApply = true)
public class LocalDateConverter implements AttributeConverter<LocalDate, Date> {

    public Date convertToDatabaseColumn(LocalDate attr) {
        return Date.valueOf(attr);
    }

    public LocalDate convertToEntityAttribute(Date dbData) {
        return dbData.toLocalDate();
    }
}
```

Java Persistence 2.2 MR

Date/time support

- Define standard handling of Date/Time APIs
- existing solution is to use AttributeConverters
- 3rd party libraries
- https://java.net/jira/browse/JPA_SPEC-63

Java Persistence 2.2 MR

Stream support in queries

```
Query q = em.createQuery("select e from Employee e");
int total = 0;
for (Employee e: q.getResultList()) {
    total += e.getSalary();
}
```


Java Persistence 2.2 MR

Stream support in queries

```
Query q = em.createQuery("select e from Employee e");
int total = 0;
for (Employee e: q.getResultList()) {
    total += e.getSalary();
}
```

```
Query q = em.createQuery("select e from Employee e");
int total = q.getResultList().stream().collect(
    Collectors.summingInt(Employee::getSalary));
```

Java Persistence 2.2 MR

Stream support in queries

- Let Query directly return Stream object

```
Query q = em.createQuery("select e from Employee e");  
int total = q.getStreamResult().collect(  
    Collectors.summingInt(Employee::getSalary));
```

Java Persistence 2.2 MR

Stream support in queries

- Let Query directly return Stream object

```
Query q = em.createQuery("select e from Employee e");  
q.setHint("stream.result.type", "parallel");  
int total = q.getStreamResult().collect(  
    Collectors.summingInt(Employee::getSalary));
```

Java Persistence 2.2 MR

Query string from CriteriaQuery

- Get the JPQL/SQL String Representations for a Criteria Query

ORM Framework	Query implementation	How to get the JPQL String	How to get the SQL String
EclipseLink	JpaQuery	getDatabaseQuery().getJPQLString()	getDatabaseQuery().getSQLString()
Hibernate	Query	N/A	getQueryString()
OpenJPA	QueryImpl	getQueryString()	N/A

Java Persistence 2.2 MR

Query string from CriteriaQuery

```
assertEquals(  
    "SELECT b FROM Book b ORDER BY b.id DESC",  
    findAllBooksCriteriaQuery.unwrap(JpaQuery.class)  
        .getDatabaseQuery().getJPQLString());
```

```
assertEquals(  
    "SELECT b FROM Book b ORDER BY b.id DESC",  
    findAllBooksCriteriaQuery.getJPQLString());
```

```
assertEquals(  
    "SELECT b FROM Book b ORDER BY b.id DESC",  
    findAllBooksCriteriaQuery.getSQLString());
```

Java Persistence 2.2 MR

AutoCloseable









- Let EntityManager and EntityManagerFactory implement AutoCloseable

```
try (EntityManager em = emf.createEntityManager()) {
    em.getTransaction().begin();
    ...
    em.getTransaction().commit();
} catch (PersistenceException pe) {
    em.getTransaction().rollback();
}
```

Java Persistence 2.2 MR

- Add pagination support to JPQL and Criteria APIs
- Allow creating CriteriaQuery from Query
- Add subquery(EntityType) to CriteriaQuery
- Query metamodel by entity name
- NoSQL
- Multitenancy

Java Persistence 2.2 MR

Fix Version: 2.2
Order by
 JPA_SPEC-115 add @Repeatable(containerClass.class) to all existing annotations for which there exists a container annotation
 JPA_SPEC-109 Allow AttributeConverters to be CDI injectable
 JPA_SPEC-99 Add ability to stream the result of a query execution
 JPA_SPEC-98 Improve Entity Graph annotations
 JPA_SPEC-89 Introduce scroll as alternative to getResultList
 JPA_SPEC-63 JPA next should support Java 8 Date and Time types
 JPA_SPEC-60 Upload official/standard JPA 2.1 API jar to Maven Central
 JPA_SPEC-43 Allow type level annotations to be used as meta-annotations

Call to Action - Participate!

- JPA Specification Project:
<https://java.net/projects/jpa-spec/>
- JPA Specification Issue Tracker:
https://java.net/jira/browse/JPA_SPEC
- JPA Specification mailing list:
users@jpa-spec.java.net
- Reference Implementation:
<http://eclipselink.org>

Session Surveys

Help us help you!!

- Oracle would like to invite you to take a moment to give us your session feedback. Your feedback will help us to improve your conference.
- Please be sure to add your feedback for your attended sessions by using the Mobile Survey or in Schedule Builder.

Q&A

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Thank You!

Integrated Cloud

Applications & Platform Services



JavaOne™

ORACLE®

ORACLE®