Roadmap Java Persistence API

Índice

1. Puntos destacados

- JPA es un API basado en la tecnología ORM (*Object Relational Mapping*) para trabajar con entidades persistentes
- Existen distintas implementaciones del API, en el módulo hemos trabajado con la implementación de Hibernate
- JPA es parte de la especificación EJB 3.0 de Java EE, pero también puede usarse en aplicaciones Java SE y aplicaciones web.
- JPA se basa en JDBC y es necesario tener en funcionamiento una base de datos SQL (como MySQL). Para usar el API debemos instalar el conjunto de librerías que lo implementan y el fichero de configuración `META-INF/persistence.xml`.
- Una de las características fundamentales de JPA es su simplicidad. Muchas opciones de configuración se definen por defecto.
- Las entidades en JPA son POJOs, clases Java sencillas a las que se les añaden una anotaciones para indicar las características de persistencia.
- En el mapeado entidad-relación las entidades se convierten en tablas y sus atributos en columnas.
- Es posible definir en JPA todas las posibles asociaciones entre entidades: uno-a-uno, uno-a-muchos, muchos-a-uno y muchos-a-muchos. El mapeo de muchas de estas asociaciones se implementa utilizando claves ajenas en la tabla propietaria de la relación. La definición de estas claves ajenas se realiza desde la entidad opuesta con el elemento `mappedBy` de la anotación que define la relación.
- El `EntityManager` es el elemento fundamental del API. Es el encargado de hacer persistentes las entidades que gestiona y de mantener la sincronización entre el contexto de persistencia (estado en memoria de las entidades) y la base de datos.
- Cuando el entity manager se cierra, todas las entidades del contexto de persistencia quedan desconectadas (*detached*) de la base de datos.
- La sincronización del contexto de persistencia y la base de datos se realiza mediante el volcado (*flush*) de sentencia SQL. Este volcado se realiza por defecto cuando se cierra una transacción o cuando se realiza una consulta.
- Por defecto, las colecciones y ciertos atributos de las entidades se cargan de forma perezosa (*lazy loading*), de forma que se guardan referencias y sólo se recuperan de la base de datos cuando se accede a ellos.
- En JPA es necesario abrir y cerrar una transacción siempre que se realiza una operación de actualización. Las transacciones que se utilizan en Java SE son las del recurso (el `autocommit` de JDBC). En Java EE se utilizan transacciones distribuidas JTA.
- En JPA se define un potente lenguaje de consulta basado en SQL llamado JPQL.

2. Certificación Sun

Los conocimientos de JPA son necesarios para superar la certificación de desarrollador de

componentes de negocio (*Sun Certified Business Component Developer (SCBCD)*), que es la siguiente nivel una vez obtenida la certificación de desarrollador de componentes web.

La página oficial con los requisitos de la certificación se puede encontrar en el siguiente enlace: Sun Certified Business Component Developer (SCBCD). En concreto, se listan los siguientes objetivos:

Entidades JPA

- Identify correct and incorrect statements or examples about the characteristics of Java Persistence entities.
- Develop code to create valid entity classes, including the use of fields and properties, admissible types, and embeddable classes.
- Identify correct and incorrect statements or examples about primary keys and entity identity, including the use of compound primary keys.
- Implement association relationships using persistence entities, including the following associations: bidirectional for @OneToOne, @ManyToOne, @OneToMany, and @ManyToMany; unidirectional for @OneToOne, @ManyToOne, @OneToMany, and @ManyToMany.
- Given a set of requirements and entity classes choose and implement an appropriate object-relational mapping for association relationships.
- Given a set of requirements and entity classes, choose and implement an appropriate inheritance hierarchy strategy and/or an appropriate mapping strategy.
- Describe the use of annotations and XML mapping files, individually and in combination, for object-relational mapping.

Operaciones sobre las entidades

- Describe how to manage entities, including using the EntityManager API and the cascade option.
- Identify correct and incorrect statements or examples about entity instance lifecycle, including the new, managed, detached, and removed states.
- Identify correct and incorrect statements or examples about EntityManager operations for managing an instance's state, including eager/lazy fetching, handling detached entities, and merging detached entities.
- Identify correct and incorrect statements or examples about Entity Listeners and Callback Methods, including: @PrePersist, @PostPersist, @PreRemove, @PostRemove, @PreUpdate, @PostUpdate, and @PostLoad, and when they are invoked.
- Identify correct and incorrect statements about concurrency, including how it is managed through the use of @Version attributes and optimistic locking.

Unidades y contextos de persistencia

- Identify correct and incorrect statements or examples about JTA and resource-local entity managers.

- Identify correct and incorrect statements or examples about container-managed persistence contexts.
- Identify correct and incorrect statements or examples about application-managed persistence contexts.
- Identify correct and incorrect statements or examples about transaction management for persistence contexts, including persistence context propagation, the use of the EntityManager.joinTransaction() method, and the EntityTransaction API.
- Identify correct and incorrect statements or examples about persistence units, how persistence units are packaged, and the use of the persistence.xml file.
- Identify correct and incorrect statements or examples about the effect of persistence exceptions on transactions and persistence contexts.

JPQL

- Develop queries that use the SELECT clause to determine query results, including the use of entity types, use of aggregates, and returning multiple values.
- Develop queries that use Java Persistence Query Language syntax for defining the domain of a query using JOIN clauses, IN, and prefetching.
- Use the WHERE clause to restrict query results using conditional expressions, including the use of literals, path expressions, named and positional parameters, logical operators, the following expressions (and their NOT options): BETWEEN, IN, LIKE, NULL, EMPTY, MEMBER [OF], EXISTS, ALL, ANY, SOME, and functional expressions.
- Develop Java Persistence Query Language statements that update a set of entities using UPDATE/SET and DELETE FROM.
- Declare and use named queries, dynamic queries, and SQL (native) queries.
- Obtain javax.persistence.Query objects and use the javax.persistence.Query API.

Hemos cubierto bastantes de estos objetivos en el módulo. Muchos de los que no han sido cubiertos aquí los trataremos en el módulo de EJB.

3. Recursos adicionales

3.1. Bibliografía

- Mike Keith, Merrick Schincariol, *Pro EJB 3. Java Persistence API*, Apress, 2006
- Christian Bauer, Gavin King, *Java Persistence with Hibernate*, Manning, 2006

3.2. Enlaces

Referencias

- Paquete javax.persistence
- Manual de referencia de Hibernate-JPA

- Dali - Entorno IDE para JPA incluido en Eclipse

Tutoriales y artículos

- Using the Java Persistence API in Desktop Applications
- Dali Object-Relational Mapping Tool
- Artículo en TheServerSide: Defining your object model with JPA
- Blog de Gavin King
- Presentación de Martin Krajcy