Java Persistence API

Narzędzia i Aplikacje Java EE

Michał Piotrowski
Michał Wójcik
michal.wojcik@eti.pg.gda.pl
http://mwojcik.eti.pg.gda.pl

Katedra Architektury Systemów Komputerowych
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

7 października 2014

Java Persistence API

JPA — Java Persistence API:

- mechanizm pozwalający na odwzorowanie obiektów języka Java na tabele w relacyjnej bazie danych,
- wykorzystywany w:
  - Java SE,
  - Java EE (uproszczone używanie i konfigurowanie):
    - warstwa webowa,
    - warstwa EJB;
- nie wymaga tworzenia skomplikowanych DAO (Data Access Objects),
- wspomaga transakcje,
- niezależny od dostawcy bazy danych,
- możliwość uniknięcia zabawy z SQL.

Dostęp do bazy danych

Dostęp do bazy danych:

- Połączenie z bazą danych jest zasobem udostępnianym przez serwer aplikacji pod określonym adresem, np.: jdbc/baza,
- z poziomu aplikacji połączenia do bazy danych uzyskujemy tak samo, jak inne zasoby udostępniane przez serwer aplikacji, tj.:
  - przez adnotację,
  - korzystając z JNDI;
- konfiguracja parametrów dostępu do bazy danych (adres serwera bazy danych, nazwa użytkownika, hasło) jest zależna od serwera aplikacji.

Klasy encyjne

Klasy encyjne (entity classes) — klasy przechowywane w bazie danych:

- zwykłe klasy POJO (Plain Old Java Object),
- pola klasy nie mogą być publiczne,
- każdy obiekt encyjny ma jednoznacznie identyfikujący go klucz główny
  - złożone klucze główne są reprezentowane przez oddzielne klasy, które muszą implementować metody `hashCode()` i `equals()`,
- oznaczone za pomocą adnotacji.

Klasy encyjne – adnotacje

Wybrane adnotacje w JPA:

- `@Entity` — oznaczenie klasy encyjnej,
- `@Table` — oznaczenie tabeli:
  - `name` — nazwa tabeli w bazie danych;
- `@Column` — oznaczenie kolumny:
  - `name` — nazwa kolumny w bazie danych,
  - `nullable` — ustawienie czy pole jest wymagane;
- `@JoinColumn` — mapowanie kolumn przy łączeniu encji:
  - `name` — nazwa kolumny z kluczem zapożyczonym,
  - `referencedColumnName` — nazwa kolumny na którą wskazuje klucz,
- `@Id` — klucz główny,
- `@GeneratedValue` — autogenerowalny klucz,
- `@Transient` — nie podlega utrwaleniu,
- `@Temporal` — wymagane dla typów `Date` i `Calendar`,

Klasy encyjne – adnotacje relacji

Adnotacje powiązań pomiędzy encjami:

- `@OneToOne`:
  - `cascade` — operacje na elementach zależnych od siebie,
  - `mappedBy` — ustala ustala pole lub własność będące właścicielem relacji;
- `@OneToMany`:
  - `cascade`,
  - `mappedBy`;
- `@ManyToOne`:
  - `cascade`;
- `@ManyToMany`:
  - `cascade`,
  - `mappedBy`.

Rodzaje powiązań:

- jednokierunkowe — tylko jedna encja ma pole wskazujące na drugą,
- dwukierunkowe — obie encje mają pola wskazujące na siebie nawzajem.

Klasy encyjne – dziedziczenie

Dziedziczenie w klasach encyjnych:

- klasa bazowa jako encja:
  - pojedyncza tabela,
  - osobna tabela dla każdej klasy,
  - osobna tabela dla dodatkowych własności;
- klasa bazowa nie jest encją, ale opisuje jej elementy.

Persistence Unit

Persistence Unit — zbiór encji w aplikacji:

- `persistence.xml` — plik konfiguracyjny, w katalogu `META-INF`,
- wybrane elementy:
  - `persistence-unit` — opis konkretnego PU:
    - `name` — nazwa,
    - `transaction-type` — rodzaj transakcji: `RESOURCE-LOCAL`, `JTA`;
  - `provider` — dostawca implementacji,
  - `class` — nazwa klasy encyjnej wraz z pakietem,
  - `property` — konfiguracja połączenia z bazą (zwłaszcza JAVA SE).

Entity Manager

Entity Manager — dostarcza metod operowania na transakcjach i encjach:

- persistence context — grupa unikalnych encji zarządzanych podczas działania aplikacji,
- persistent identity — posiada unikatowy identyfikator,
- stany encji:
  - new — nie jest jeszcze związana z persistence context, nie posiada persistent identity,
  - managed — związana z persistence context, posiada persistent identity,
  - detached — nie jest już związana z persistence context, posiada persistent identity,
  - removed — związana z persistence context, posiada persistent identity, oznaczona do usunięcia;
- `EntityManagerFactory` — udostępnia dostęp do zarządcy encji (w Java SE), może być wykorzystana przez kilka wątków,
- nie może być wykorzystywany przez więcej niż jeden wątek.

Entity Manager

Dostęp do zarządcy encji:

- z poziomu klas zarządzanych przez kontener uzyskujemy przez adnotację `@PersistenceContext EntityManager em`,
- z poziomu pozostały klas uzyskujemy korzystając z JNDI,
- z poziomu aplikacji nie uruchomionej na serwerze aplikacji poprzez ręczne stworzenie fabryki.

Zarządzanie encjami

Zarządzanie encjami:

- `EntityManager`:
  - `void persist(Object o)` — dodanie nowej encji (new -> managed),
  - `EntityTransaction getTransaction()` — zwraca obiekt transakcji,
  - `<T> T merge(T entity)` — synchronizuje stan encji z bazą,
  - `void remove(Object o)` — usunięcie encji (managed -> removed),
  - `void refresh(Object o)` — aktualizuje stan encji na podstawie bazy,
  - `<T> T find(Class<T> entityClass, Object key)` — wyszukuje na podstawie klucza prywatnego;
- `EntityTransaction`:
  - `begin()` — rozpoczyna transakcję,
  - `commit()` — kończy transakcje, zapisuje zmiany do bazy,
  - `rollback()` — usuwa transakcję z kolejki, odłącza zarządzane encje.

Zapytania

Inne zapytania mogą być tworzone jako:

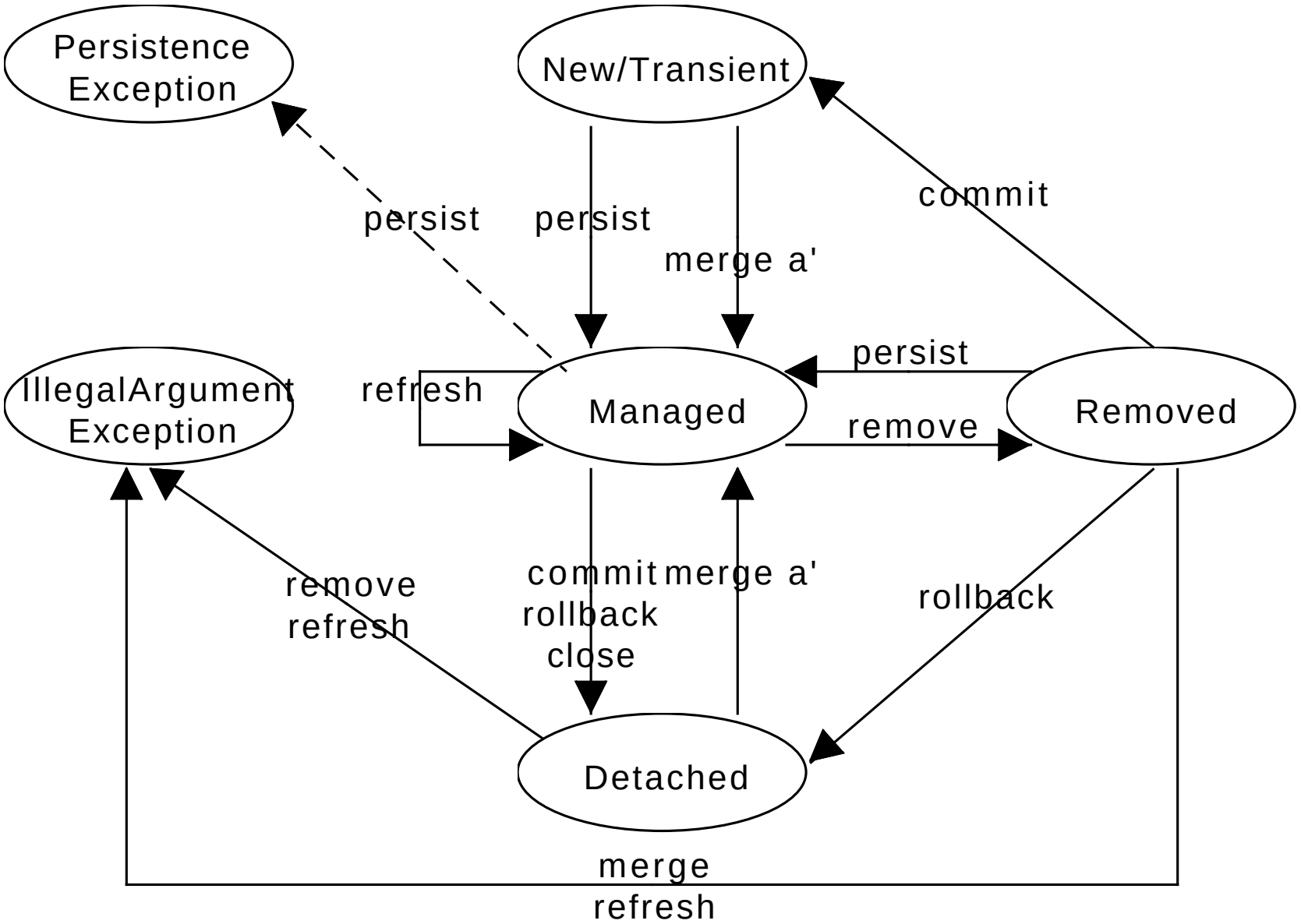
- Java Persistence Query Language,
- SQL wykorzystywanej bazy danych (tzw. zapytanie natywne),
- wykorzystując *Criteria API*.

Zapytania wywołuje się korzystając z zarządcy encji

Można predefiniować zapytania (tzw. named query)

- nazwy muszą być unikalne w danej jednostce trwałości,
- zasięg nazw ograniczony jest przez jednostkę trwałości.

Zarządzanie encjami



Transakcje zarządzane przez beana

Transakcje zarządzane przez beana:

- możliwość korzystania z:
- JDBC API dostarcza metody wywoływane na połączenie z bazą danych:
  - `setAutoCommit()`,
  - `commit()`,
  - `rollback()`;
- JPA API dostarcza obiekt transakcji dostarczający metod:
  - `begin()`,
  - `commit()`,
  - `rollback()`;
- dwa rodzaje transakcji:
  - `EntityTransaction` — uzyskana z zarządcy encji (np.: Java SE),
  - `UserTransaction` — transakcja użytkownika uzyskana za pomocą adnotacji.

Transakcje

Uwagi:

- w konfiguracji serwera aplikacji można ustawić:
  - limit czasu trwania transakcji,
  - poziom izolacji transakcji;
- Transakcje kontrolowane przez Java EE (obsługiwane przez kontener lub korzystające z JTA API) pozwalają na:
  - modyfikacje danych w kilku bazach danych wewnątrz jednej transakcji,
  - przeprowadzanie operacji na dwóch serwerach aplikacji w jednej transakcji.

Java DB

Java DB — wspierana przez SUN (Oracle) dystrybucja Apache Derby.
Apache Derby:

- mała baza danych, około 2.6 MB,
- w całości zaimplementowana w języku Java,
- dystrybuowana na licencji Apache License, Version 2.0,
- oparta na standardach JDBC i SQL,
- wpiera także tryb klient/serwer.

Pomocne Źródła I

[1] *The Java EE 7 Tutorial.*
Oracle, January 2013.
http://download.oracle.com/javaee/7/tutorial/doc/.