JAVA PERSISTENCE (REVIEW)

Niko Ibrahim, MIT

Pemrograman Web Lanjut

Latar Belakang

- Aplikasi komputer selalu terdiri dari:
  - Logika bisnis
  - Interaksi dengan sistem lain
  - Antarmuka
  - …dan peyimpanan data (*persistence*)
- Data yang diproses di dalam aplikasi biasanya disimpan dalam suatu tempat yang disebut: DATABASE
- DATABASE sangat penting karena:
  - Menyimpan data bisnis
  - Bertindak sebagai titik pusat antar aplikasi
  - Memproses data melalui trigger dan stored procedure
- Saat ini hampir semua database aplikasi menggunakan RELATIONAL DATABASES sebagai engine penyimpanan data
- RELATIONAL DATABASE:
  - menyimpan data dalam bentuk ROWS dan COLUMNS
  - Data diidentifikasi dengan menggunakan PRIMARY KEY
  - Hubungan antar tabel menggunakan FOREIGN KEYS dan JOINT TABLES

DUNIA JAVA dan DATABASE

- MASALAHNYA: sstilah-istilah yang ada di dalam RELATIONAL DATABASE benar-benar tidak dikenal di dunia Java yang merupakan bahasa berorientasi objek!!
- Di dalam Java, kita memanipulasi OBJECT yang merupakan instance dari CLASS.
- OBJECT:
  - Merupakan turunan (*interitance*) dari OBJECT lainnya
  - Memiliki *reference* kepada OBJECT lainnya
- Di dunia JAVA dikenal istilah-istilah: CONCRETE CLASS, ABSTRACT CLASS, INTERFACE, ENUMERATION, ANNOTATION, METHOD, ATTRIBUTE, dll
- OBJECT hanya dapat diakses lewat Java Virtual Machine (JVM). Apabila JVM tidak jalan, maka OBJECT pun akan menghilang.
- Seringkali OBJECT itu perlu disimpan (*persist*) dalam arti disimpan secara permanen dalam suatu tempat penyimpanan (harddisk, flash memory, dll)
- OBJECT yang dapat disimpan untuk digunakan di lain waktu itulah yang disebut sebagai PERSISTENT OBJECT.

CARA MENYIMPAN DATA PADA JAVA java.io.serializable

- Ada beberapa cara untuk menyimpan (*persist*) OBJECT di dalam Java.
- Salah satu cara adalah dengan menggunakan mekanisme SERIALIZATION, yaitu proses mengkonversi suatu OBJECT menjadi untaian (sequence) yang berisi BITS.
- OBJECT dapat di-*serialize* pada disk, melalui jaringan/internet, dalam format yang independen sehingga dapat digunakan dimanapun bahkan di sistem operasi yang berbeda.
- Mekanisme serializeable ini di atur oleh interface JAVA.IO.SERIALIZABLE
- Namun masih banyak kelemahan dari interface ini, misalnya: tidak mendukung bahasa query

JDBC: populer dan siap pensiun

- Cara lain untuk menyimpan OBJECT adalah dengan menggunakan Java Database Connectivity (JDBC)
- JDBC merupakan Application Programming Interface (API) standar yang digunakan untuk mengakses RELATIONAL DATABASE.
- JDBC memiliki kemampuan untuk melakukan koneksi pada database, mengeksekusi statement SQL, dan menerima hasil Query.
- Walaupun JDBC masih secara luas digunakan, namun popularitasnya semakin menurun dan digantikan dengan *tools* yang lebih *powerful* yaitu Objek Relational Mapping (ORM)

PRINSIP DASAR dan TOOLS ORM

- Pada prinsipnya, ORM mendelegasikan akses RELATIONAL DATABASE melalui *tools* atau *frameworks* yang pada akhirnya akan menjadikan RELATIONAL DATABASE memiliki 'rasa' atau view dalam bentuk OBJECT ORIENTED dan demikian pula sebaliknya.
- TOOLS ORM memungkinkan terjadinya pemetaan BIDIRECTIONAL antara DATABASE dan OBJECT.
- Berikut ini beberapa tools (frameworks) yang merupakan implementasi dari ORM: Hibernate, TopLink, EclipseLink, Java Data Object (JDO), dan Java Persistence API (JPA).
- JPA merupakan teknologi yang paling populer karena merupakan paket bawaan dari Java EE 6.

Pengertian
Java Persistence API (JPA)

- JPA merupakan abstraksi tingkat lanjut dari JDBC yang memungkinkan aplikasi terbebas dari bahasa SQL.
- Semua CLASS dan ANNOTATIONS dari JPA berada dalam package JAVAX.PERSISTENCE
- Komponen utama JPA adalah sebagai berikut:
  - ORM, yang merupakan mekanisme untuk memetakan OBJECT untuk disimpan di dalam RELATIONAL DATABASE
  - ENTITY MANAGER API, untuk melakukan operasi-operasi yang berhubungan dengan DATABASE seperti CREATE, READ/RETRIEVE, UPDATE, dan DELETE (CRUD). Dengan mengunakan API ini, kita terbebas dari JDBC API maupun SQL.
  - Java Persistence Query Language (JPQL), yang berfungsi untuk mengambil (*retrieve*) data dengan menggunakan bahasa query berorientasi object.
  - Mekanisme TRANSACTION dan LOCKING pada saat mengakses data secara simultan (bersamaan) dengan menggunakan Java Transaction API (JTA)
  - CALLBACK dan LISTENER untuk menguhubungkan logika bisnis yang ada di dalam aplikasi dengan LIFECYCLE dari PERSISTENT OBJECT

Pengertian ENTITAS
Apa bedanya dengan OBJECT biasa?

- Pada saat kita melakukan MAPPING OBJECT ke dalam RELATIONAL DATABASE, maka OBJECT itu disebut dengan istilah ENTITAS.
- OBJECT adalah *instance* yang hanya hidup di memory.
- ENTITAS adalah object yang hidup sebentar di memory, dan disimpan (*persist*) di dalam DATABASE.
  - Memiliki kemampuan untuk dipetakan ke dalam DATABASE
  - Dapat berupa CONCRETE maupun ABSTRACT class
  - Dapat memiliki INHERITANCE, RELATIONSHIP, dll
- Setelah ENTITAS dipetakan, maka selanjutnya dapat dikelola oleh JPA
  - Dapat didimpan ke dalam database
  - Dapat dihapus dari database
  - Dapat di-query menggunakan JPQL

Bagaimana cara JPA memetakan OBJECT ke dalam DATABASE?

- Melalui METADATA
- Di dalam setiap ENTITAS akan diberikan METADATA yang menjelaskan PEMETAAN yang akan dilakukan.
- METADATA inilah yang akan memungkinkan *TOOLS/FRAMEWORK* mengenali suatu ENTITAS dan menginterpretasikan PEMETAAN yang akan dilakukan.
- METADATA ini dapat ditulis dengan 2 cara:
  - ANNOTATIONS: kode program di dalam ENTITAS langsung dianotasi (ditandai) dengan menggunakan beberapa anotasi (tanda) yang ada di dalam package JAVAX.PERSISTENCE
  - XML DESCRIPTORS: pemetaan didefinisikan di dalam sebuah file XML yang akan di-deploy (disimpan ke server) bersamaan dengan ENTITAS. Teknik kedua ini lebih bermanfaat apabila sering terjadi perubahan konfigurasi database.

CONTOH ENTITAS "BOOK"

```
@Entity
public class Book {
    @Id @GeneratedValue
    private Long id;

    @Column(nullable = false)
    private String title;

    private Float price;

    @Column(length = 2000)
    private String description;

    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;

    // Constructors, getters, setters
}
```

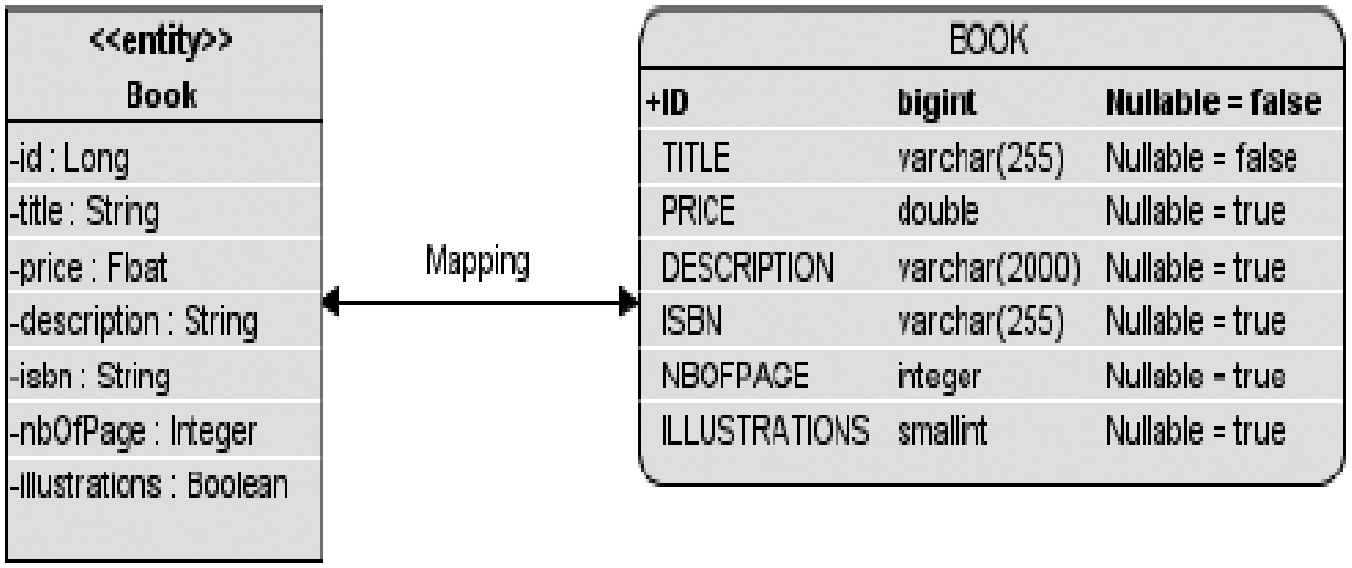
PENJELASAN ENTITAS “BOOK”

- Untuk dikenali sebagai ENTITAS, class BOOK harus dianotasi dengan menggunakan anotasi “@Entity”
- Untuk menandai atribut sebagai PRIMARY KEY, kita gunakan anotasi “@Id”
- Untuk membuat AUTOMATIC INCREMENT VALUE pada PRIMARY KEY, kita gunakan anotasi “@GeneratedValue”
- Untuk meng-customize keadaan default pada pemetaan kolom, kita gunakan anotasi “@Column”, misalnya:
  - “TITLE” dibuat NOT-NULL

    ```
    @Column(nullable = false)
    ```
  - “DESCRIPTION” memilki panjang 2.000 karakter

    ```
    @Column(length = 2000)
    ```
- Engine persistence (JPA) kemudian akan memetakan ENTITAS “BOOK” tersebut menjadi TABEL “BOOK” pada DATABASE

PEMETAAN
ENTITAS BOOK ←→ TABEL BOOK



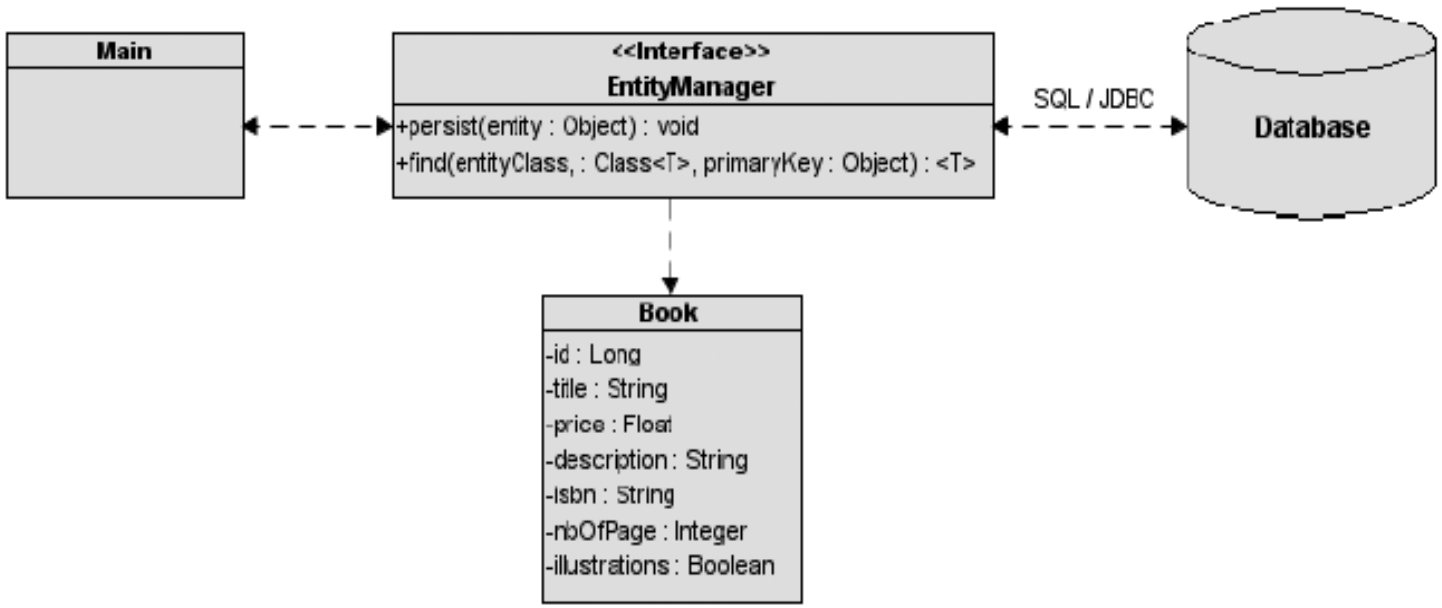
Meng-QUERY Entitas

- JPA memiki kemampuan untuk meng-query entitas melalui cara berorientasi objek tanpa melibatkan foreign key atau kolom pada database (artinya: tanpa melibatkan SQL)
- Otak Utama yang mengatur ENTITAS ini adalah ENTITY MANAGER
- ENTITY MANAGER berfungsi:
  - Mengatur ENTITAS (membaca dan menuliskannya ke dalam DATABASE)
  - Mengoperasikan CRUD pada ENTITAS
  - Membuat QUERY yang kompleks dengan menggunakan JPQL
- Secar teknis-nya, ENTITY MANAGER merupakan implementasi dari interface yang disediakan oleh engine ORM, seperti EclipseLink (yang dipakai di NetBeans secara default)
- Berikut ini potongan kode yang menunjukkan cara membuat ENTITY MANAGER dan menyimpan (*persist*) ENTITAS "BUKU":

```
EntityManagerFactory emf =
   Persistence.createEntityManagerFactory("BookStorePU");
EntityManager em = emf.createEntityManager();
em.persist(book);
```

WHOLE PICTURE:
EntityManager, Aplikasi (*Main Class*), Entitas, JDBC, dan Database

- Gambar ini memperlihatkan bagaimana interface EntityManager dapat digunakan oleh suatu class (Main class) untuk memanipulasi ENTITAS (BOOK).
- Dengan menggunakan methods seperti persist() dan find(), maka ENTITY MANAGER sukses menyembunyikan kerumitan JDBC pada saat mengakses DATABASE, termasuk pada saat menjalankan perintah SQL INSERT dan SELECT



JPQL
Java Persistence Query Language

- Entity Manager memungkinkan kita untuk meng-query ENTITAS.
- QUERY di sini mirip dengan query SQL biasa. Namun, QUERY ini tidak dilakukan terhadap DATABASE, melainkan terhadap ENTITAS.
- Query terhadap ENTITAS ini dilakukan dengan menggunakan JPQL.
- SINTAKS-nya merupakan CAMPURAN antara sintaks SQL dan sintaks bahasa berorientasi objek yaitu menggunakan tanda TITIK (.)
- Contoh: untuk mengambil data buku dengan judul "Laskar Pelangi"

```
SELECT b FROM Book b WHERE b.title = 'Laskar Pelangi'
```

- Statement JPQL dapat dieksekusi dengan menggunakan DYNAMIC QUERY (dibuat pada saat runtime), STATIC QUERY (didefinisikan pada saat compile time) atau bahkan dapat menggunakan NATIVE SQL apabila diperlukan. (NATIVE SQL = SQL biasa yg ada di database)
- STATIC QUERY biasa disebut juga NAMED QUERY dapat didefinisikan menggunakan ANNOTATION maupun XML.

Contoh STATIC/NAMED QUERY *findBookByTitle*

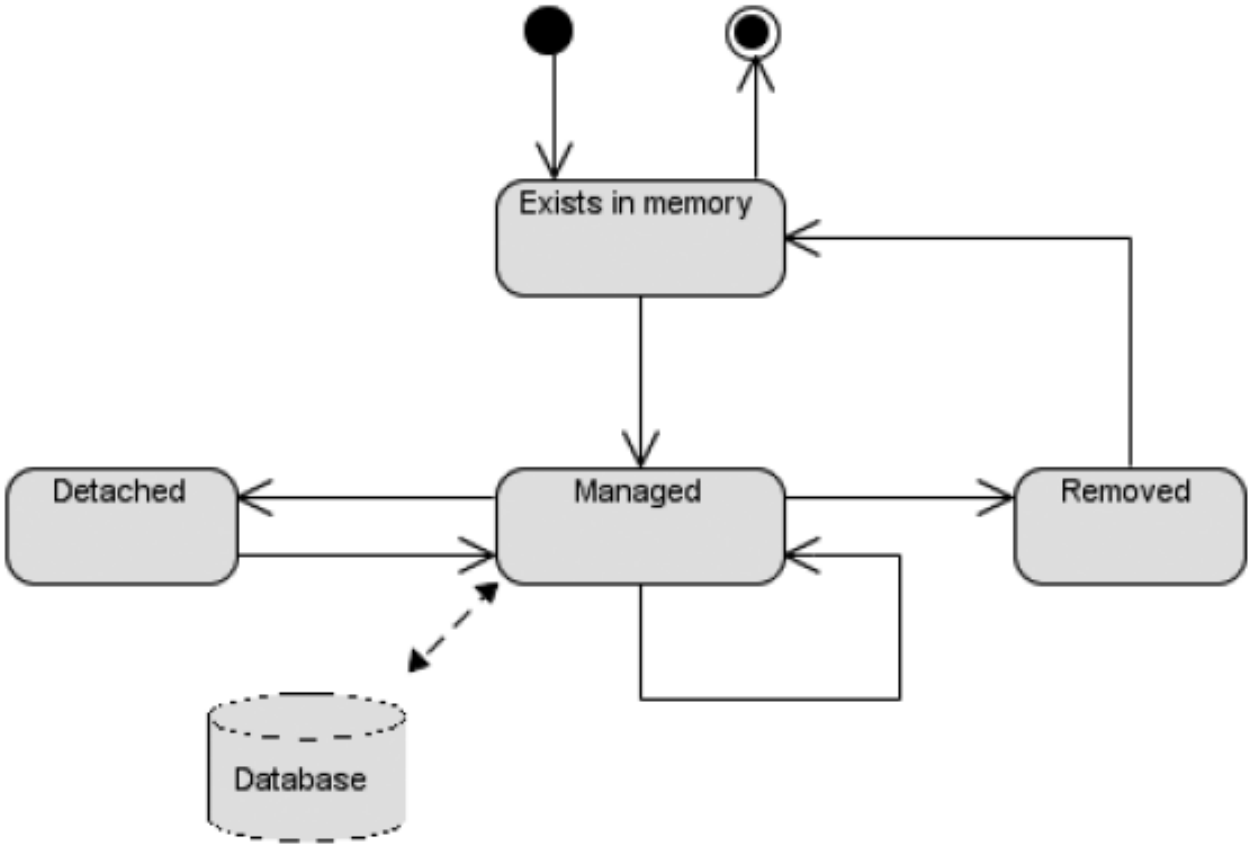
```
@Entity
@NamedQuery(
    name = "findBookByTitle",
    query = "SELECT b FROM Book b WHERE b.title ='Laskar Pelangi'")
public class Book {
    @Id @GeneratedValue
    private Long id;

    @Column(nullable = false)
    private String title;
    private Float price;
    @Column(length = 2000)
    private String description;
    private String isbn;
    private Integer nbOfPage;
    private Boolean illustrations;
    // Constructors, getters, setters
}
```

LIFE CYLE Entitas

- ENTITAS pada dasarnya hanyalah Class Java biasa (istilahnya: Plain Old Java Object / POJO) yang di-managed ataupun tidak di-managed oleh ENTITY MANAGER.
  - Pada saat di-managed (attached), ENTITAS memiliki identitas persistence dan state-nya di-sinkronisasi dengan DATABASE.
  - Pada saat tidak di-managed (detached), ENTITAS dapat digunakan selayaknya class biasa pada Java.
- Hal ini berarti bahwa ENTITAS memiliki suatu LIFE CYLE
  - Pada saat kita membuat intance dari ENTITAS BOOK, maka OBJECT akan berada di memory. Saat itni, JPA tidak tahu menahu sama sekali mengenai OBJECT tersebut
  - Pad saat ENTITAS menjadi MANAGED oleh ENTITY MANAGER, maka state-nya dipetakan dan di-sinkronisasi dengan TABEL BOOK.
  - Pada saat dipanggil perintah EntityManager.remove(), maka data akan dihapus dari DATABASE, namun OBJECT JAVA-nya tetap hidup di memory (sampai dihapus dengan sendirinya oleh *garbage collector*)

Life Cycle Entitas (illustrated)



CallBack dan Listener

- Operasional terhadap ENTITAS dikategorikan menjadi 4:
  - PERSISTING
  - UPDATING
  - REMOVING
  - LOADING
- Ke-empat operasi tsb berkorespondensi langsung dengan operasi di DATABASE yaitu:
  - INSERTING
  - UPDATING
  - DELETING
  - SELECTING
- Setiap operasi memiliki event “PRE” dan “POST” (kecuali loading hanya memiliki event “POST”) misalnya @PrePersist dan @PostPersist
- EVENT ini dapat dimanfaatkan oleh Entity Manager untuk mengakses *business method*.
- JPA memungkinkan kita untuk menghubungkan *businsess logic* pada ENTITAS pada saat EVENTS tsb terjadi
- Anotasi yang diset pada method ENTITAS disebut CALLBACK methods
- Anotasi yang diset pada class eksternal disebut LISTENER
- Sebagai analogi, kita dapat memandang method CALLBACK dan LISTENER ini sebagai TRIGGERS pada Relational Database.

TUTORIAL

- Kerjakan Tutorial JPA dan MySQL
- Kumpulkan hari ini juga