Beyond Java ORM with Versant JPA (Part 1)

German Viscuso
Developer Relations Manager
Versant Corporation
March 2012

Kembali pada tahun 2004 Sun Microsystems berjuang dengan kekurangan yang Enterprise Jawa Kacang (EJB) 2.0 dan EJB 2.1 spesifikasi. Itu jelas pada waktu itu penyederhanaan arsitektur, standardisasi, dan ketekunan yang diperlukan untuk memudahkan kehidupan Jawa pengembang.
Dengan demikian Jawa ketekunan API 1,0 spesifikasi lahir pada tahun 2006 sebagai bagian dari EJB 3.0 (JSR- 220) dengan lebih lanjut perangkat tambahan di 2009: JPA 2.0 (JSR-317). Itu diterima standar yang termasuk dukungan untuk banyak fitur yang EJB pengembang yang meminta, termasuk dukungan untuk objek peningkatan pemodelan, warisan, Polimorfisme, diperluas query language, dan kaya metadata untuk spesifikasi object relational mapping.

Namun prestasi terbesarnya JPA's tidak hanya untuk membakukan ORM ketekunan teknologi untuk Java developers tetapi juga untuk menggabungkan praktek-praktek terbaik untuk ringan berbasis POJO ketekunan (menyingkirkan arsitektur entitas kacang tua dan kelas berat).

Spesifikasi JPA menarik pada ide-ide, konsep dan standar dari ketekunan terkemuka kerangka seperti TopLink, hibernasi dan Jawa Data objek (JDO), serta pada sebelumnya EJB berhasil wadah ketekunan. Tapi JPA itu sendiri adalah hanya sebuah spesifikasi; set antarmuka, dan membutuhkan pelaksanaan dan database untuk bertahan untuk.
Di sini adalah tempat Versant datang ke dalam bermain. Tim Versant ditinjau negara-of-the-art dari JPA teknologi untuk membawa Anda Versant JPA, penyedia JPA 2.0 untuk Versant Database teknologi.

Dalam artikel ini multi-bagian saya akan memperkenalkan blok bangunan dasar Jawa ketekunan api sebagai saya memandu Anda melalui pelaksanaan Versant JPA dari perspektif pengguna (developer).

Entitas Entitas pada dasarnya adalah sebuah objek ringan gigih domain. Bentuk entitas didefinisikan dalam ketekunan mampu kelas (alias entitas kelas dengan biasanya menambahkan anotasi @Entity (lihat) Catatan 1):

JPA Entity Class—Person

```
/* Copyright (C) 2011 Versant Inc.  http://www.versant.com */

package com.versant.jpa.tutorial.model;

import javax.persistence.*;

@Entity
public class Person {

    @Id
    private long id;

    private String firstName;

    private String lastName;

    private Person() {

    }

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    @Override
    public String toString() {
        return firstName + " " + lastName;
    }

}
```

Gambar 1: kelas entitas JPA adalah kelas Java biasa dengan penambahan @Entity anotasi

Secara tradisional keadaan objek tersebut akan dimasukkan ke sebuah tabel dalam database relasional. Sementara contoh dari suatu entitas akan sesuai dengan individu baris dalam tabel. The
informasi pemetaan yang diperlukan harus ditambahkan ke dalam campuran (atau gagal oleh JPA pelaksanaan) untuk mencocokkan objek domain ke meja. Dapatkah Anda bayangkan menyingkirkan semua overhead relasional sementara masih mampu menggunakan JPA?

Masukkan Versant JPA.

Catatan bahwa dalam contoh orang di atas kami hanya perlu dua anotasi: @Entity dan @Id. @Id penjelasan menunjuk lapangan untuk digunakan untuk entitas objek identitas di datastore (atau primary key di relasional lingo). Itu adalah benar-benar semua yang Anda butuhkan untuk membuat ini menggunakan kelas entitas Versant JPA. Karena Versant toko entitas kelas contoh sebagai objek, ada tambahan anotasi menentukan pemetaan informasi yang diperlukan.

Unit ketekunan

Unit ketekunan JPA adalah pengelompokan logis pengguna didefinisikan persistable kelas dengan terkait pengaturan seperti koneksi database:

persistence.xml

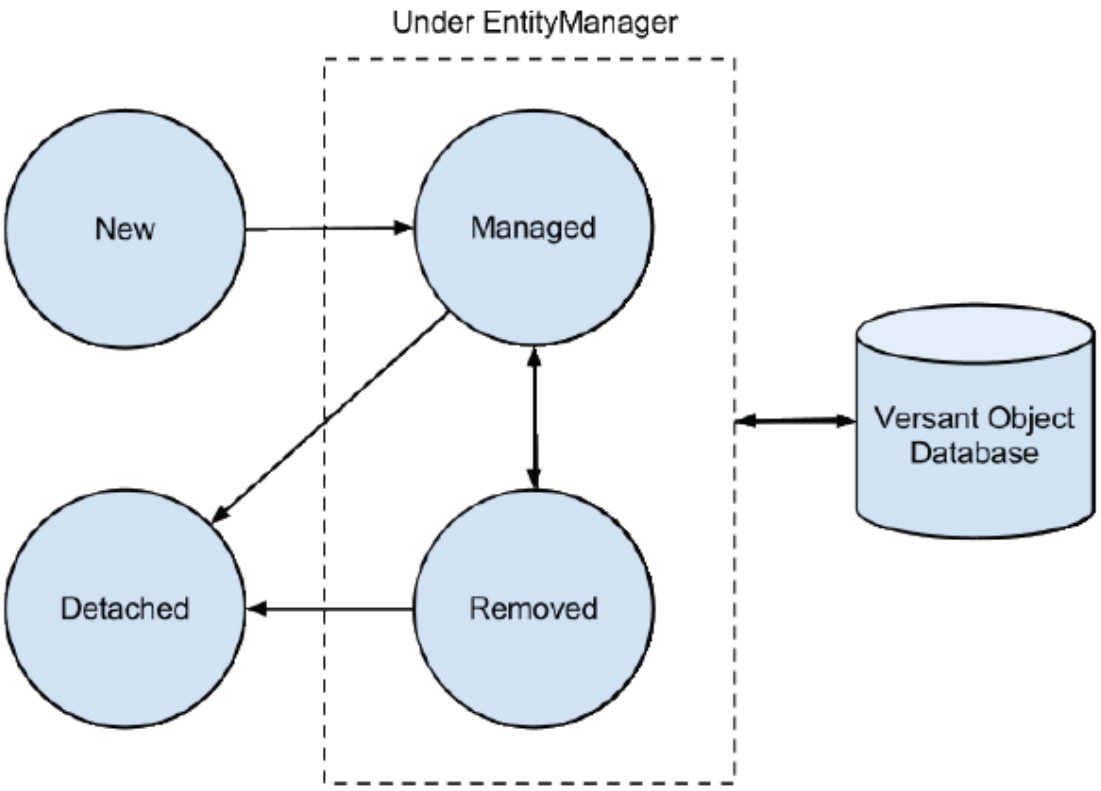
```
<persistence version="2.0">
    <persistence-unit name="jpa_tutorial_persistence_unit" transaction-type="RESOURCE_LOCAL">
        <class>com.versant.jpa.tutorial.model.Person</class>
        <class>com.versant.jpa.tutorial.model.Book</class>
        <properties>
            <property name="versant.connectionURL" value="jpa_tutorial@localhost" />
        </properties>
    </persistence-unit>
</persistence>
```

Gambar 2: unit ketekunan JPA

Unit ketekunan daftar kelas mampu ketekunan (yaitu entitas kelas) yang membuat model data aplikasi Anda. Ini adalah elemen-elemen yang dapat Anda lihat dalam Persistence.xml file di atas. Contoh kelas-kelas ini akan disimpan dalam database didefinisikan oleh pengaturan sambungan.
Catatan: Unit ketekunan didefinisikan dalam sebuah file XML yang bernama persistence.xml yang terletak di aplikasi META_INF direktori.

Manajer entitas

Dengan JPA, objek kelas entitas dalam aplikasi Anda dikelola oleh EntityManager. An entitas manajer pabrik bertanggung jawab untuk entitas manajer. Ini adalah pabrik yang terkait dengan unit kegigihan melalui properti koneksi ke Versant database.
Pada dasarnya, antarmuka EntityManager menyediakan API untuk berinteraksi dengan entitas. Entitas dapat di negara yang berbeda yang berhubungan erat dengan EntityManager Layanan:



Gambar 3: Entity siklus hidup di JPA

Entitas dibuat untuk pertama kalinya berada dalam keadaan yang disebut "New" dan mereka bukan bagian dari Database atau EntityManager apapun menyadari hal itu. Ketika EntityManager yang melakukan "bertahan" operasi lebih dari entitas negara entitas perubahan "Managed". Hal ini harus dilakukan secara aktif transaksi (lihat Transaksi bawah) yang setelah komit efektif akan menyimpan entitas dalam database.

Entitas diambil dari database (misalnya melalui query) juga dalam keadaan "Managed" jadi ketika mereka dimodifikasi dalam transaksi perubahan akan disimpan ke database pada berikutnya komit.

The "menghapus" operasi memungkinkan EntityManager untuk menandai suatu entitas untuk penghapusan (perubahan negara dari "Dikelola" untuk "Dihapus" dan entitas akan dihapus dari database pada komit).

Akhirnya, entitas dapat benar-benar terputus dari EntityManager sehingga mereka mencapai negara dari "Terpisah" (entitas misalnya pergi ke negara ini ketika EntityManager yang ditutup).

Layanan yang paling penting dari API EntityManager adalah:

- bertahan - Menyimpan entitas baru.
- merge - Update keadaan entitas ke dalam database.
- melepaskan - disassociates objek dari database.

● menghapus - Menghapus contoh entitas.

Jadi, bagaimana saya mulai bekerja dengan EntityManager di Versant JPA?

The first thing is to create a factory and tell it which persistence unit to use for its configuration:
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("jpa_tutorial_persist~ ence_unit");

The resulting factory can now be called on to provide you with an EntityManager:
EntityManager em = emf.createEntityManager();

Transaksi
Semua modifikasi pada objek persisten, termasuk membuat yang baru, yang dibuat dalam transaksi. Sebuah transaksi adalah unit kerja yang dilakukan oleh klien, aplikasi Anda, dan database server.
EntityManager bertanggung jawab untuk transaksi. Anda bisa mendapatkan akses ke transaksi dengan manajer getTransaction () metode. Anda bekerja dengan transaksi dengan menandai awal dan titik akhir dengan *begin()* and *commit()*:
em.getTransaction().begin();
// do stuff with persistent objects . . .
em.getTransaction().commit();

Setiap perubahan pada objek entitas dalam transaksi ditulis ke database (committed) ketika Anda menelepon commit () metode. Apakah ada kontingensi bukan commit (), Anda dapat menghubungi rollback () metode untuk membatalkan setiap perubahan yang Anda buat. Ketika Anda membuat sebuah instance baru dari kelas entitas, EntityManager tidak tahu apakah Misalnya harus gigih atau tidak. Jadi, Anda harus memberitahu EntityManager yang Anda ingin Misalnya untuk bertahan dalam database dengan memanggil manajer entitas *persist* () metode untuk objek baru misalnya (sehingga objek Anda pada akhirnya masuk ke negara "Dikelola").

Queries

Menyimpan benda tidak berharga jika Anda tidak dapat menemukan mereka lagi. JPA menggunakan Query Language JPA (JPQL) yang dapat dianggap sebagai versi berorientasi objek dari SQL. Akrab dengan SQL Pengguna harus menemukan JPQL sangat mudah dipelajari dan digunakan.

Sebuah permintaan dilakukan dengan meminta EntityManager untuk contoh Query. Permintaan yang diinginkan kriteria, predikat, disediakan untuk manajer CreateQuery () metode.

Query query = em.createQuery("select p from Person p");

Predikat query (dinyatakan sebagai string yang mengikuti sintaks JPQL) hanya memilih semua anggota dari class Person entitas. Permintaan dikirim ke server V / OD dan hasilnya Koleksi dikembalikan:

List<Person> resultList = query.getResultList();

Kita kemudian dapat menggunakan loop untuk mencetak hasil:

for (Person person : resultList) {

System.out.println(person);

}

atau sebaliknya, untuk mengeluarkan benda Orang semua dari database:

for (Person person : resultList) {

em.remove(person);

}

Seperti yang anda lihat itu sangat mudah dan lurus ke depan untuk memulai dengan Versant JPA. Jika Anda memiliki pengalaman sebelumnya dengan Java Persistence API Anda akan merasa seperti di rumah. Dalam bagian 2 dari seri ini saya akan menunjukkan lebih banyak fitur canggih seperti Cascading ketekunan (persistence-by-reachability), pertanyaan lebih lanjut, ubah pelacakan dan penggabungan terpisah benda. Jika Anda ingin mencoba contoh kerja dari semua fitur yang dibahas di sini silahkan pergi ke depan dancobalah Preview JPA Teknis Versant ini yang tersedia sebagai percobaan gratis:

http://community.versant.com/jpa.aspx
Catatan 1: Secara umum semua penjelasan JPA dapat digantikan dengan mendefinisikan tag di unit ketekunan itu file XML ORM.