Bab 4

Berkenalan dengan Hibernate sebagai solusi Model dalam MVC

4.1 Tujuan

Dalam bab ini, kita akan mendiskusikan bagaimana Hibernate berperan sebagai Model dalam MVC pattern. Kita akan mendiskusikan perkembangan Object Relational Mapping secara umum, sekaligus mencoba mengimplementasikan Hibernate sebagai relational mapping.

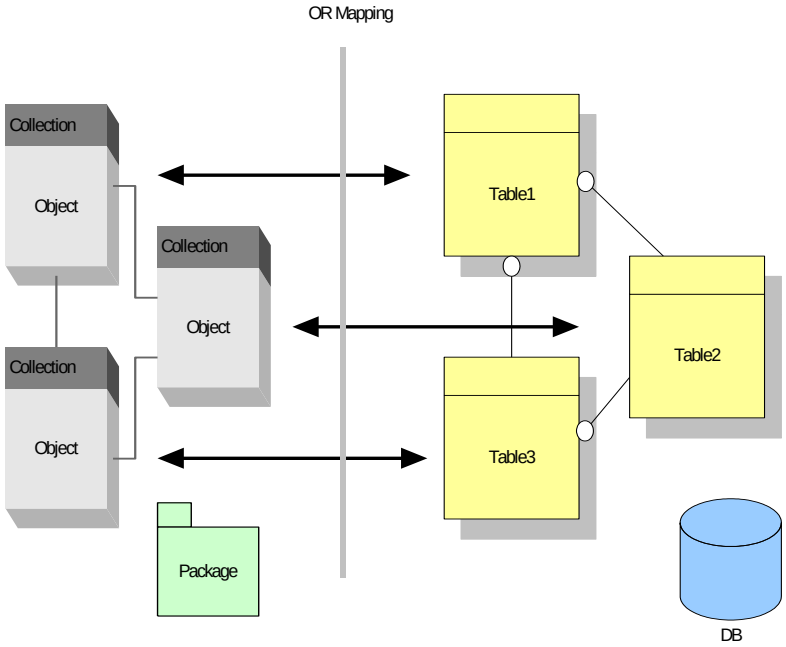
Pada akhir bab ini, pelajar diharapkan dapat:

- Memahami peran dari Hibernate sebagi Model dalam MVC
- Mengenal perkembangan teknologi model dalam MVC
- Mengetahui tool-tool yang diperlukan untuk men-generate Hibernate
- Mengimplementasikan relational object mapping dengan Hibernate
- Memahami peran Hibernate Query Language (HQL)
- Mengimplementasikan relationship table dengan object mapping

4.2 Teknologi Model pada MVC

Model adalah sebuah layer yang lebih dekat ke sumber data, baik itu berupa database, webservices, atau file system. Untuk membuat model ini dalam berinteraksi dengan Controller, dapat dilakukan dengan menggunakan mekanisme membuat thread baru dengan New, atau melakukan injection. Bab berikutnya akan membahas bagaimana injection, yang mana akan membuat layer controller dan model menjadi sebuah satu kesatuan.

Bab ini akan membahas lebih dalam mengenai mekanisme model yang berinteraksi dengan database, yang sering disebut teknologi ORM atau singkatan dari Object Relational Mapping. Sebenarnya teknologi ORM ada beberapa macam, seperti Toplink dari Oracle, Kodo dari Bea Systems, dan tentu saja yang akan dibahas lebih dalam dibab ini adalah Hibernate. ORM paling populer didunia.



Mekanisme Object Relational Mapping Bekerja

ORM sebenarnya adalah sebuah teknologi yang menjadi satu lapis antara aplikasi dengan database, yang mana ORM ini bekerja seperti database juga, tetapi hanya berbentuk objek. Setiap objek didalam ORM, umumnya mewakili table dalam database. Akibat dari teknologi mapping ini membuat sebuah aplikasi yang dikembangkan dengan ORM, tidak terikat dengan database manapun. Sedangkan untuk melakukan query database, dengan ORM digunakan sebuah object relational language, yang bekerja mirip dengan SQL. Umunya setiap table dan objek POJO dimap dengan sebuah XML .

Setelah spesifikasi final EJB3 ini keluar, yaitu dengan datangnya standar Java Persistence API atau JPA sebagai salah satu dari spesifikasi inti dari EJB3. Dimana sebenarnya JPA ini adalah versi anonasi dari ORM metadata. ORM mengganti XMLnya menjadi perintah anonasi didalam Java. Akibat dari standar EJB3 yang bernomor JSR 220 ini, telah membuat teknologi Hibernate, Toplink dan Kodo, yang semuanya adalah teknologi ORM, menjadi inti dari EJB3. Untuk Hibernate, agar Hibernate dapat menjadi EJB3 harus melakukan integrasi antara Hibernate core, Hibernate annonation dan Hibernate entity manager. Dengan hadirnya JPA, membuat teknologi TopLink dan Hibernate secara kasat mata adalah sama.

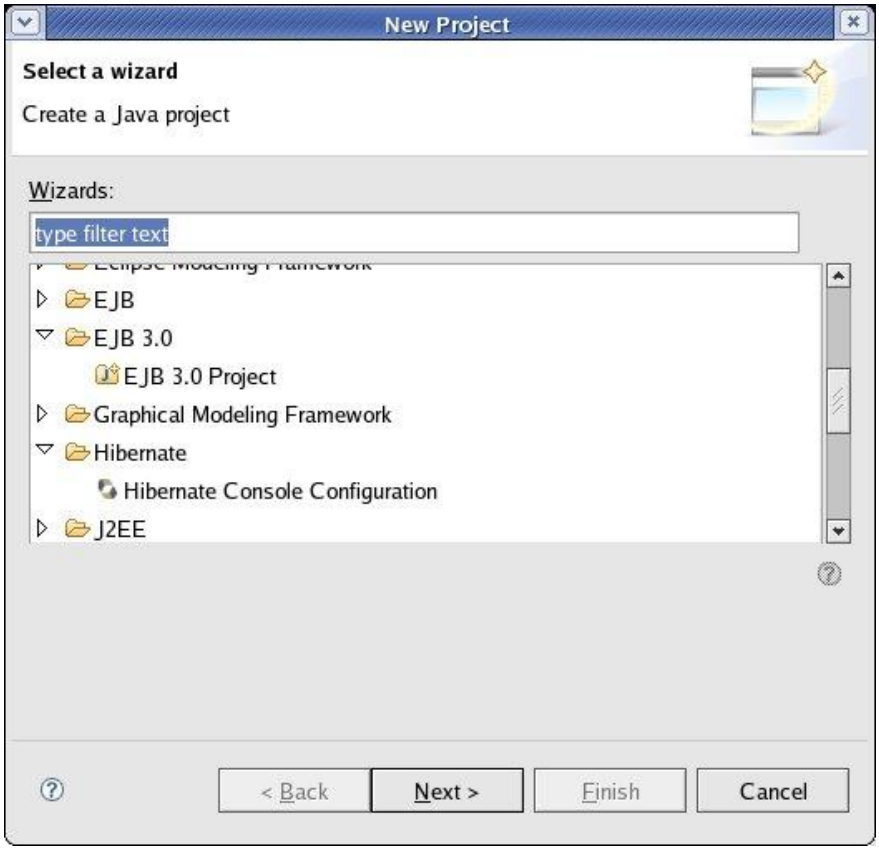
Hibernate yang datang dengan 2 versi yaitu versi core dan versi annonation, sebenarnya secara fungsi adalah sama, perbedaannya Hibernate versi core memerlukan XML sebagai mappernya. Yang mana dalam implementasinya sebenarnya hanya menganti object Configuration menjadi AnnonationConfiguration.

4.3 Berkenalan dengan Hibernate Tools

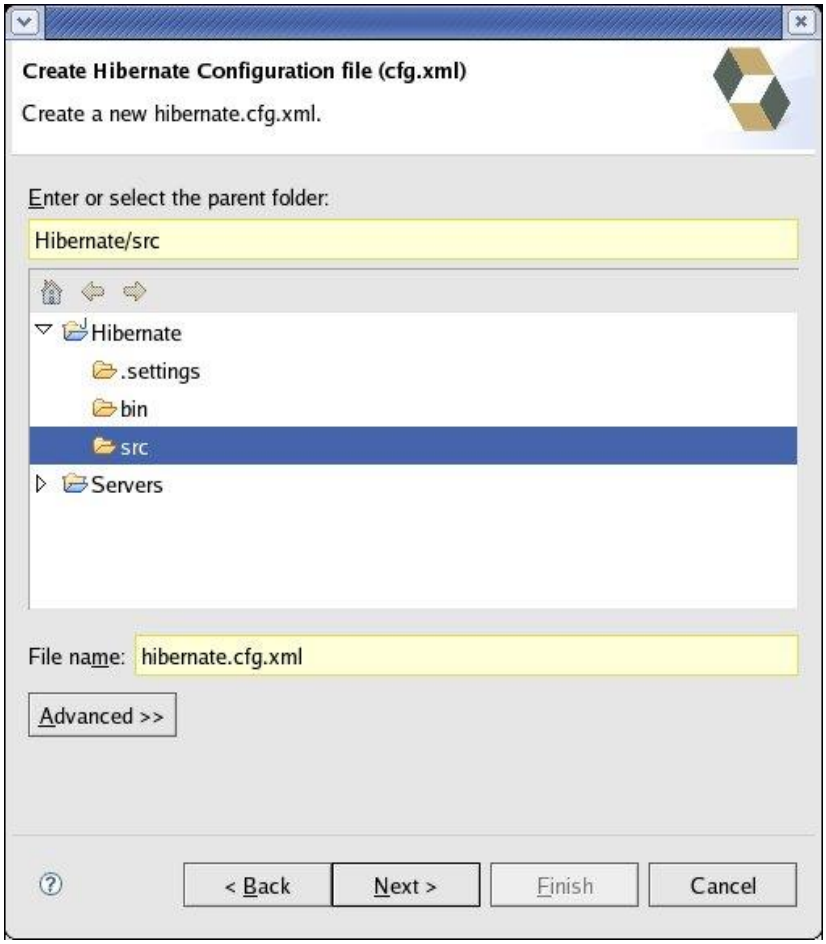
Untuk memulai sebuah projek berbasis Hibernate, yang termudah adalah dengan menggunakan Eclipse Calisto, dan menambahkan JBoss IDE kedalamnya.

Hibernate Tools sebenarnya datang dengan 2 teknologi, sebagai plugins eclipse dan sebagai script pada ant. Penulis merasakan bilamana menggunakan plugins, seorang programmer dapat mencoba HQLnya terlebih dahulu, sedangkan dengan Ant, memungkinkan diintegrasikan dengan script lainnya didalam Ant, seperti DBUnit, Junit, ataupun CVS update. DBUnit dan HibernateTool akan dibahas dalam bab ini.

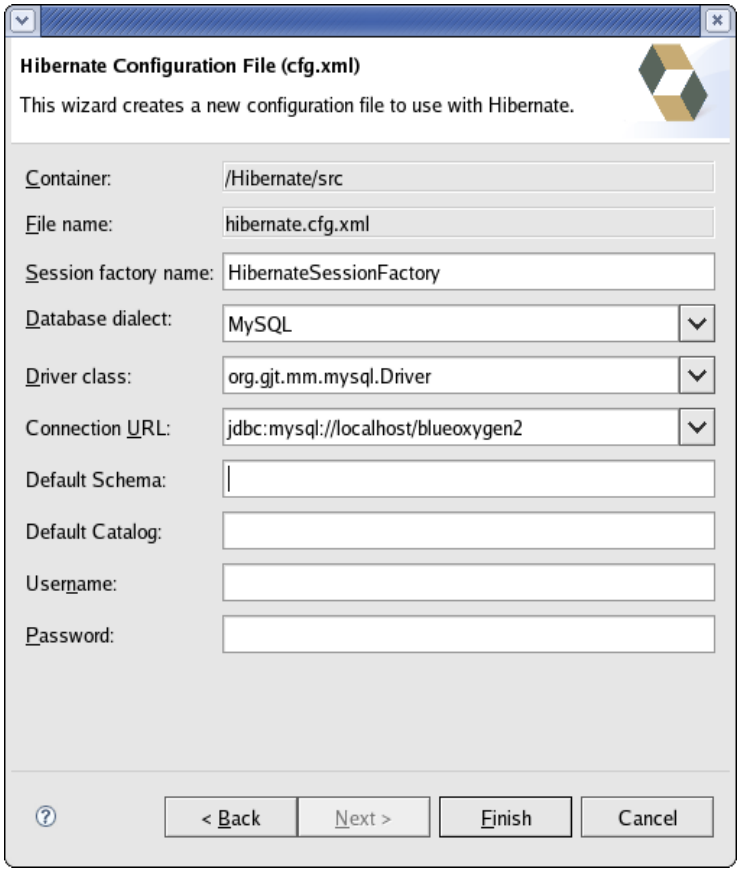
Mekanisme mengidentifikasi Hibernate Tools telah berjalan dengan baik adalah melakukan New Project, coba scroll, pasti ada kata Hibernate didalamnya. Sedangkan cara menginstallnya adalah dengan mengextract Jboss IDE atau Hibernate Tools IDE kedalam folder Eclipse. Hal ini terjadi untuk semua tipe Eclipse tanpa perlu perubahaan. Tentu saja plugins JBoss terbaru memerlukan Calisto.



Buatlah sebuah Java Project, kemudian buat sebuah Hibernate Configuration file (hibernate.cfg.xml).

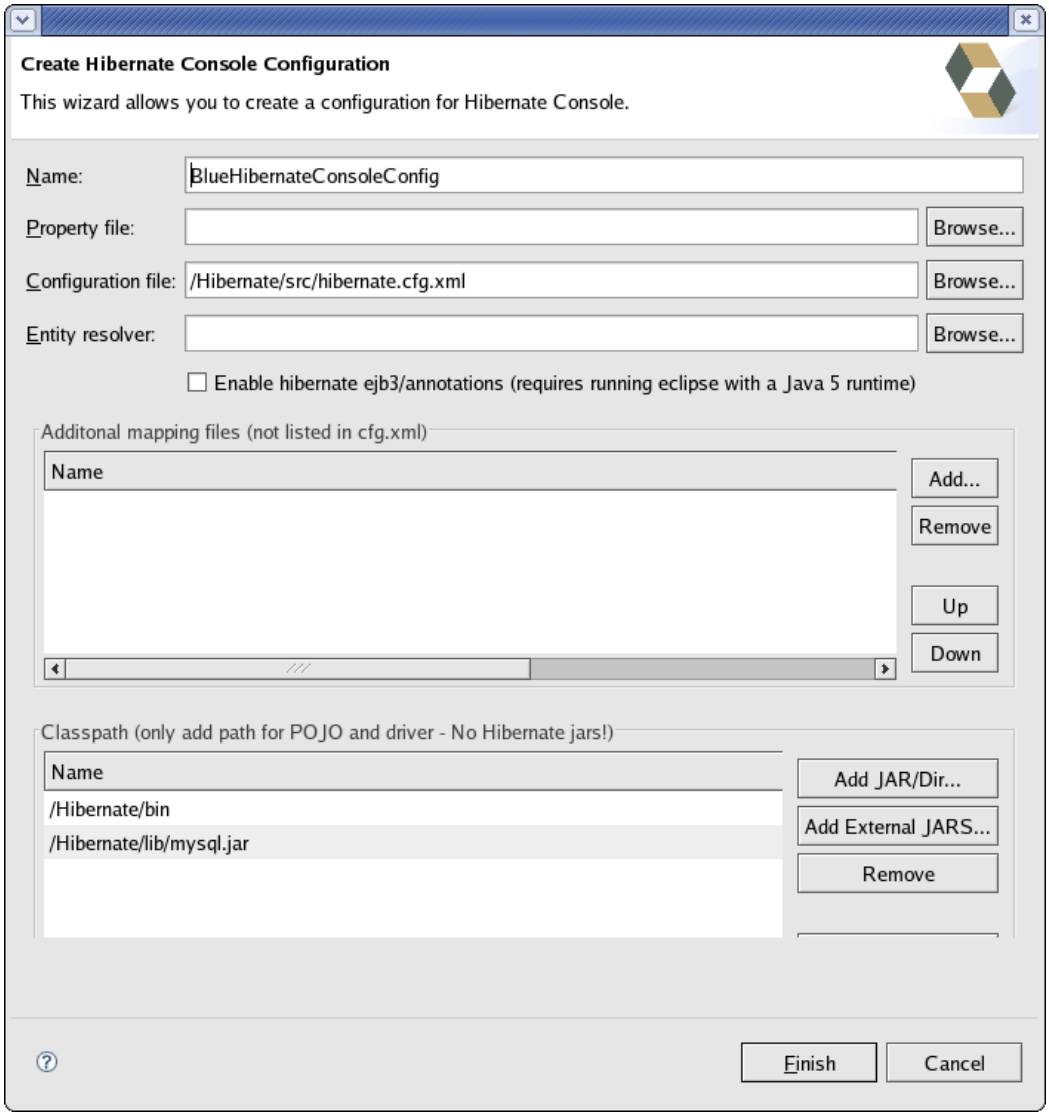


Sebaiknya hibernate.cfg.xml disimpan didalam folder src, yang artinya sama dengan dikenal saat terjadi runtime, dengan Eclipse, akan tercopy ke folder build.

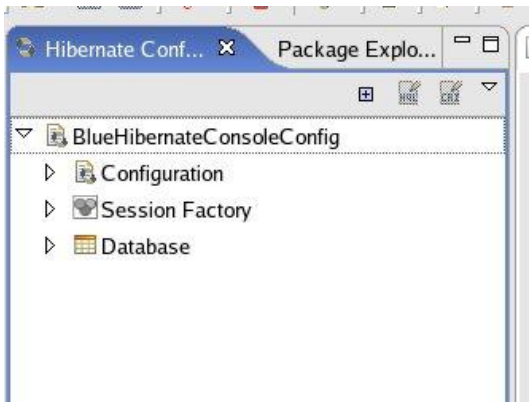



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
                "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
                "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
   <session-factory name="HibernateSessionFactory">
      <property
name="hibernate.connection.driver_class">org.gjt.mm.mysql.Driver</property>
      <property name="hibernate.connection.password">tulalit</property>
      <property
name="hibernate.connection.url">jdbc:mysql://localhost/blueoxygen2</property>
      <property name="hibernate.connection.username">root</property>
      <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
   </session-factory>
</hibernate-configuration>
```

Langkah berikutnya adalah dengan membuat Hibernate Console Configuration, configuration ini diperlukan untuk membuat HQL (Hibernate Query Language) yaitu SQLnya Hibernate, atau mentest Criteria (filter terhadap sebuah query).



Jangan lupa untuk memasukan folder binary kedalam classpath dari Hibernate Console Configuration ini. Hal ini diperlukan karena Hibernate Console ini tidak dapat mengenai objek ORM. Bilamana proses ini telah selesai dapat masuk ke Hibernate perspective, maka akan muncul sebuah tree dari Hibernate yang isinya adalah Configuration, SessionFactory dan Database. Adapun Database ini terdiri dari semua POJO yang memiliki anonasi JPA atau XML meta.

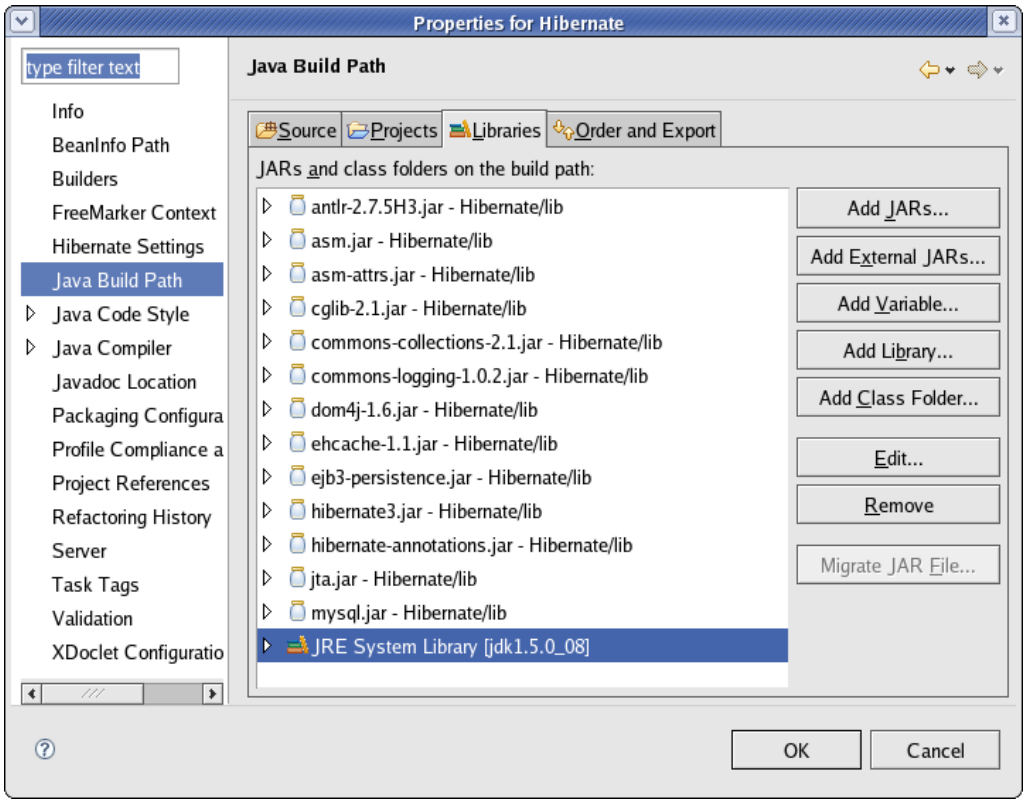


Bilaman step ini telah berhasil, artinya Hibernate plugins pada Eclipse Calisto telah terinstall dengan baik.

4.4 Membuat Object Mapping Pertama

Untuk mencoba mengimplementasikan Hibernate, terlebih dahulu buatlah sebuah POJO yang memiliki anonasi, POJO ini akan terhubung dengan table "businesspartner".

Adapun sebelum memulai langkah berikutnya, copylah sebuah .jar dari binary hibernate (dalam folder lib) yang didownload dari websitenya yaitu http://www.hibernate.org. Masukan semua jar tersebut dalam Build Pathnya Eclipse, caranya dengan mengklik kanan dari projek Javanya, kemudian tambahakan jarnya.



Bilaman telah selesai, buatlah sebuah class DefaultPersistence, objek ini adalah sebuah objek yang termapping untuk menghasilkan id yang unik dari setiap kegiatan persistensi. Objek ini sangat diperlukan dalam pengembangan kedepannya, karena dengan melakukan implementasi objek ini, akan terbentuk sebuah table dengan record id, serta akan terisi otomatis bilamana dilakukan save terhadap data baru.

Adapun DefaultPersistence ini adalah:

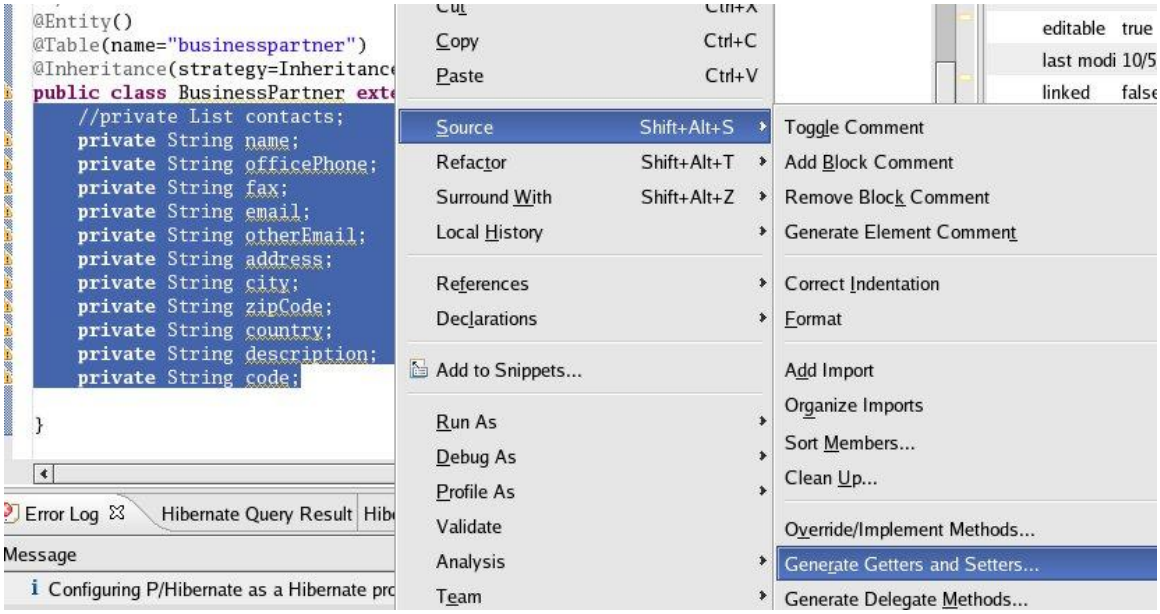
```
package org.blueoxygen.bbook2;

import java.io.Serializable;
import javax.persistence.Embedded;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.MappedSuperclass;
import org.hibernate.annotations.GenericGenerator;
```

```
@MappedSuperclass
public class DefaultPersistence implements Serializable {
    private String id;

    @Id @GeneratedValue(generator="system-uuid")
    @GenericGenerator(name="system-uuid", strategy="uuid")
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }
}
```

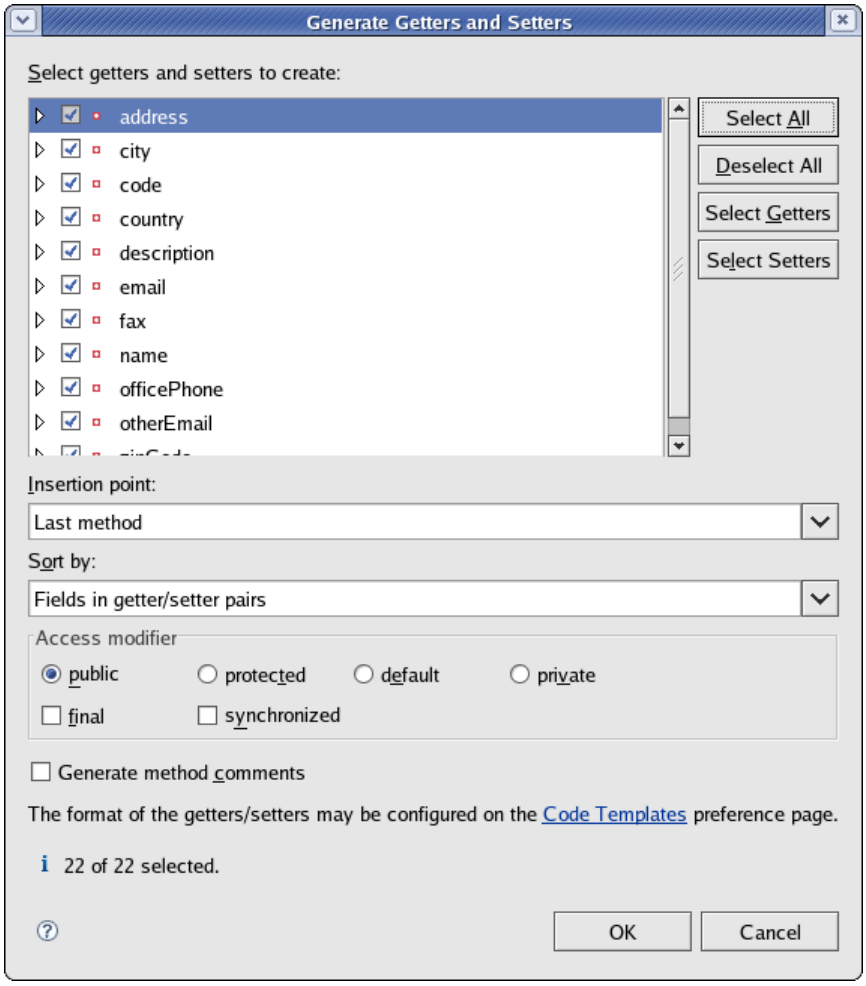
Adapun langkah yang harus dilakukan adalah dengan membuat sebuah Java class dengan nama BusinessPartner.java.

```
package org.blueoxygen.bbook2;

import java.util.ArrayList;
import java.util.List;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import org.blueoxygen.bbook2.DefaultPersistence;

import org.hibernate.Session;

@Entity()
@Table(name="businesspartner")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
public class BusinessPartner extends DefaultPersistence {
    //private List contacts;
    private String name;
    private String officePhone;
    private String fax;
    private String email;
    private String otherEmail;
    private String address;
    private String city;
    private String zipCode;
    private String country;
    private String description;
    private String code;

}
```

Setelah itu, pilih semua private object dari name sampai code, kemudian klik kanan, pilihlah



Kemudian pilihlah semua variable yang ada, dan secara otomatis semua variable akan memiliki method getXXX dan setXXX.



Tambahkan @Column() pada semua baris diatas getXXX, seperti contoh pad agetAddress():

```
@Column()
public String getAddress() {
	return address;
}
```

Setelah itu jangan lupa untuk meregister class yang telah dibuat didalam hibernate.cfg.xml, dengan menambahkan baris berikut:

```
<mapping class="org.blueoxygen.bbook2.BusinessPartner"/>
```

4.5 Mengenerate Database dengan HibernateTools

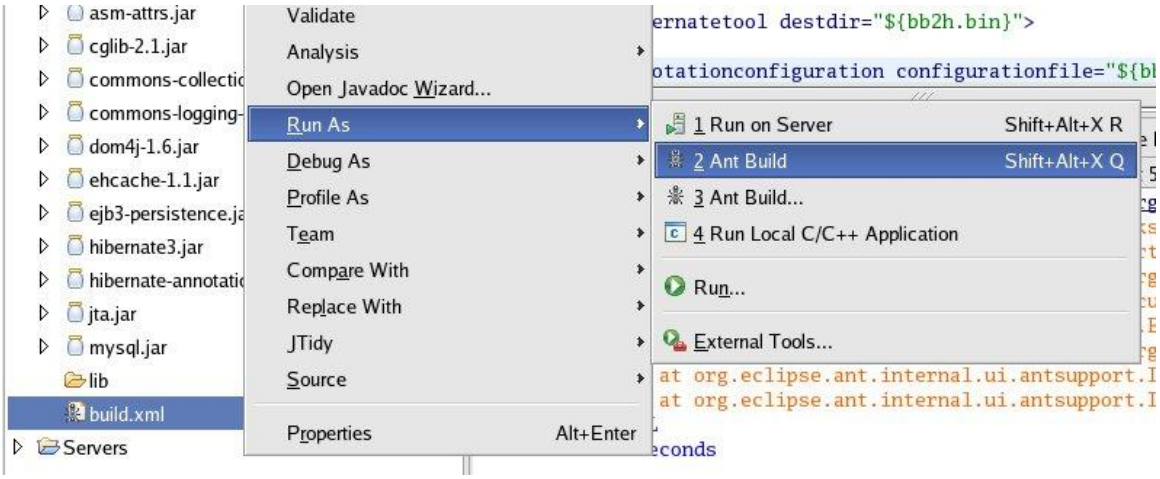
Bilamana langkah ini telah dilakukan, maka proses berikutnya adalah dengan membuat table secara otomatis menggunakan HibernateTools.

Untuk mengenerate HibernateTools diperlukan ant script (build.xml), sehingga table businesspartner dapat digenerate otomatis.

Adapun file ant atau build.xml adalah sebagai berikut:

```
<project name="bbook2hibernate" default="schemaexport">
	<target name="init">
		<property name="bb2h.base" value="/home/frans/workspace/Hibernate"/>
		<property name="bb2h.lib" value="${bb2h.base}/lib"/>
		<property name="bb2h.classes" value="${bb2h.base}/bin"/>
		<property name="bb2h.source" value="${bb2h.base}/src"/>
	</target>

	<target name="schemaexport" depends="init">
		<taskdef name="hibernatetool"
classname="org.hibernate.tool.ant.HibernateToolTask">
			<classpath>
				<fileset dir="${bb2h.lib}">
					<include name="**/*.jar"/>
				</fileset>
				<path location="${bb2h.bin}"/>
			</classpath>
		</taskdef>
		<hibernatetool destdir="${bb2h.bin}">

		<annotationconfiguration
configurationfile="${bb2h.source}/hibernate.cfg.xml"/>
		  <hbm2ddl
				drop="true"
				console="false"
				create="true"
				outputfilename="bb2h-ddl.sql"
				delimiter=";"
				export="true"
				/>
		</hibernatetool>
	</target>

</project>
```

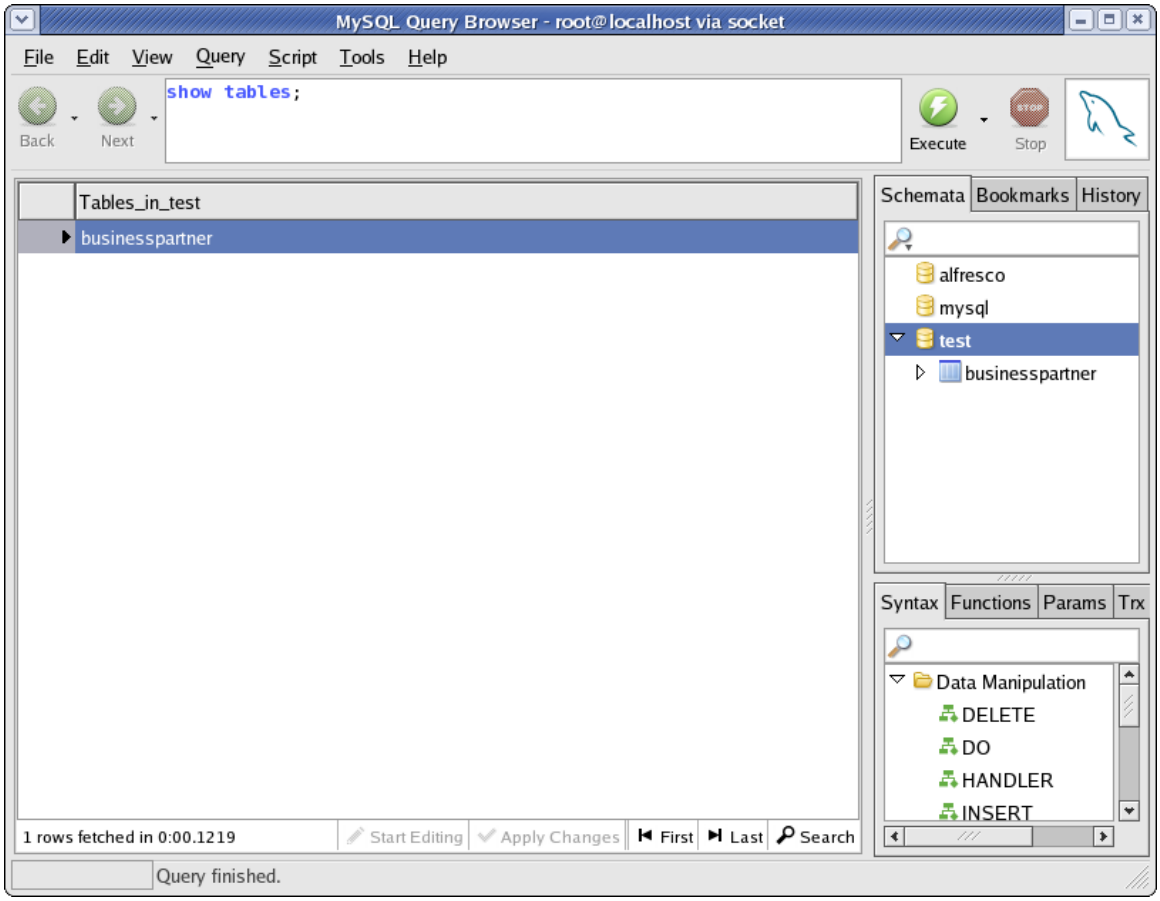
Jalankan script build.xml tersebut dengan cara melakukan klik kanan terhadap file build.xml yang telah dibuat, kemudian pilih Run As, dilanjutkan dengan Ant script.



Bilamana tidak terjadi masalah, maka console akan menghasilkan output seperti berikut ini:

```
Buildfile: /home/frans/workspace/Hibernate/build.xml
init:
schemaexport:
[hibernatetool] Executing Hibernate Tool with a Hibernate Annotation/EJB3 Configuration
[hibernatetool] 1. task: hbm2ddl (Generates database schema)
[hibernatetool] Oct 5, 2006 11:18:49 PM org.hibernate.cfg.Environment <clinit>
[hibernatetool] INFO: Hibernate 3.2 cr1
[hibernatetool] Oct 5, 2006 11:18:49 PM org.hibernate.cfg.Environment <clinit>
[hibernatetool] INFO: hibernate.properties not found
[hibernatetool] Oct 5, 2006 11:18:49 PM org.hibernate.cfg.Environment buildBytecodeProvider
[hibernatetool] INFO: Bytecode provider name : cglib
....
....
....
[hibernatetool] Oct 5, 2006 11:18:50 PM
org.hibernate.connection.DriverManagerConnectionProvider configure
[hibernatetool] INFO: connection properties: {user=root, password=****}
[hibernatetool] Oct 5, 2006 11:18:50 PM org.hibernate.tool.hbm2ddl.SchemaExport execute
[hibernatetool] INFO: schema export complete
[hibernatetool] Oct 5, 2006 11:18:50 PM
org.hibernate.connection.DriverManagerConnectionProvider close
[hibernatetool] INFO: cleaning up connection pool: jdbc:mysql://localhost/test
BUILD SUCCESSFUL
Total time: 3 seconds
```

Bilamana tidak ditemukan error, coba jalankan MySQL Query Browser, klik table test, maka secara otomatis akan terbentuk table businesspartner. Jalankan perintah "show tables", makan secara otomatis akan muncul tables_in_test bernama businesspartner. Ini artinya proses pembuatan table dari hibernate telah berhasil dengan sukses.



4.6 Menjalankan Hibernate Objek.

Langkah berikutnya bilamana pembuatan database telah berhasil, adalah dengan mencoba menjalankan sebuah aplikasi Java yang memanggil objek Hibernate.

Secara konsep proses ini adalah melakukan inisiasi HibernateSessionFactory, kemudian baru melakukan eksekusi, dimana untuk mengeksekusi diperlukan script HQL (Hibernate Query Language). Adapun script untuk melakukan testing Hibernate adalah sebagai berikut:

```
package org.blueoxygen.bbook2;

import java.util.ArrayList;
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

public class HibernateQueryTest {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                SessionFactory sf;

            Session sess;
```

```
        AnnotationConfiguration config = new AnnotationConfiguration();
        config.configure("hibernate.cfg.xml");
        sf = config.buildSessionFactory();
        sess = sf.openSession();
        BusinessPartner bpt = new BusinessPartner();
        bpt.setName("Meruvian11");
        bpt.setAddress("Tulodong Jakarta");
        sess.saveOrUpdate(bpt);
        List<BusinessPartner> bp = new ArrayList<BusinessPartner>();
        bp = sess.createQuery("FROM "+BusinessPartner.class.getName()).list();
        for(BusinessPartner business : bp){
           System.out.println("Biz Name:"+business.getName());

        }
        sf.close();
          }

}
```

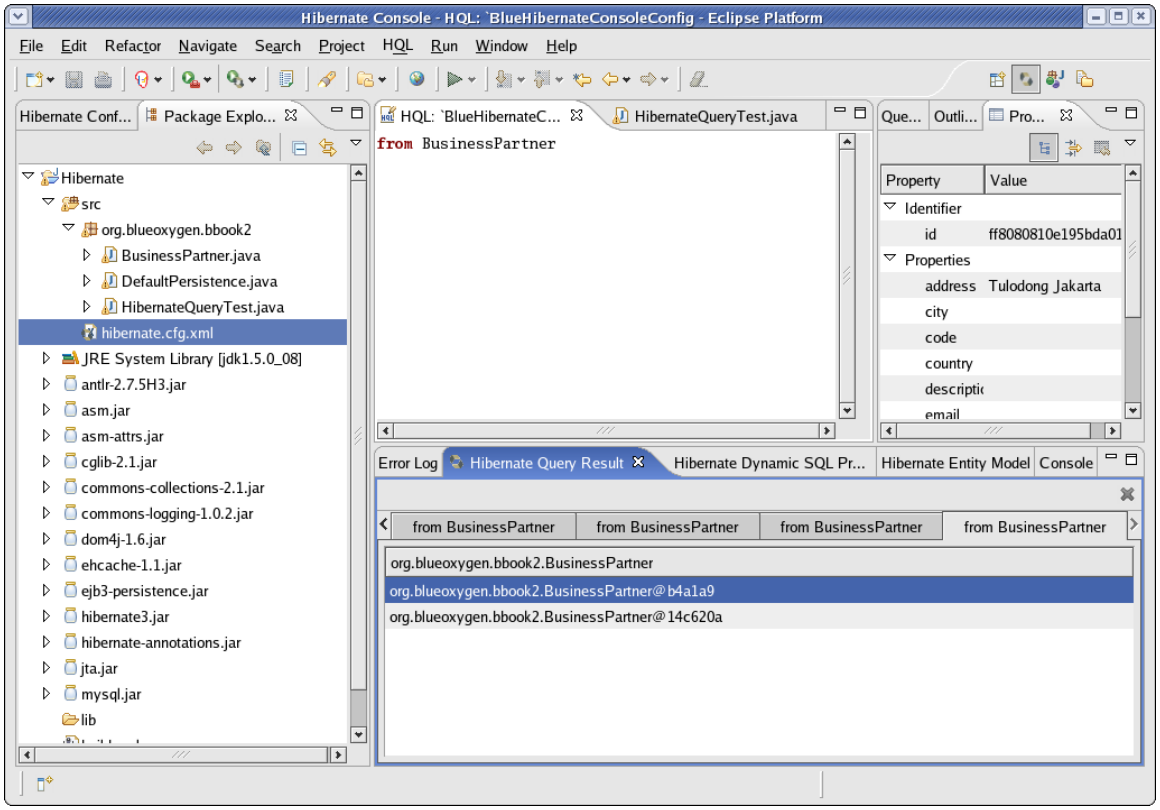
Terdapat 2 kegiatan dalam script diatas, yaitu membuat record baru dengan perintah saveOrUpdate dan melakukan query ke database. Perintah query ke database (HQL) adalah "FROM"+BusinessPartner.class.getName()).

4.7 Melakukan Query dengan HSQL Editor

Adapun cara lain untuk mengetahu total record didalam database adalah dengan menjalankan HSQL Script Editor pada Eclipse.

Cara memanggil HQL Editor adalah dengan menekan icon HSQL diatas Hibernate Configuration Config.

Masukan perintah berikut "FROM BusinessPartner", artinya query semua data dari objek BusinessPartner. Harap diperhatikan agar nama objek adalah case sensitif.



Cobalah klik Hibernate Query Result. Bilamana query telah berhasil, akan terdapat hash objek didalam Query Result, cobalah klik salah satu dari objek hash tersebut. Secara otomatis Propieties akan memunculkan informasi dari isi dari objek tersebut. Bandingkan dengan melakukan perintah "SELECT * FROM businesspartner" pada database, hasilnya harus sama.

Bilamana semua langkah diatas telah berhasil dijalankan, artinya Anda telah berhasil bekerja dengan sebuah teknologi ORM bernama Hibernate.

4.8 ORM dengan Relationship

Setelah mencoba membuat projek kecil berbasis Hibernate dengan memaping satu table dan memasukan anonasi, berikut ini adalah mengembangkan dengan metode many to many, yaitu dengan kasus Event dan Person.

Untuk lebih mudahnya buat lah dua class dengan nama Event dan Person pada projek Java yang telah ada.

Adapun kode untuk Person.java adalah

```
package org.blueoxygen.bbook2;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.ManyToMany;
```

```
import javax.persistence.Table;

import org.blueoxygen.bbook2.DefaultPersistence;

@Entity
@Table(name="person")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
public class Person extends DefaultPersistence {
	private String age;
    private String firstname;
    private String lastname;
    private Set<Event> events;

	public String getAge() {
		return age;
	}
	public void setAge(String age) {
		this.age = age;
	}
	public String getFirstname() {
		return firstname;
	}
	public void setFirstname(String firstname) {
		this.firstname = firstname;
	}
	public String getLastname() {
		return lastname;
	}
	public void setLastname(String lastname) {
		this.lastname = lastname;
	}
	@ManyToMany(mappedBy="persons")
	public Set<Event> getEvents() {
		return events;
	}
	public void setEvents(Set<Event> event) {
		this.events = event;
	}

}
```

Sedangkan kode untuk Event.java adalah

```
package org.blueoxygen.bbook2;

import java.sql.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
```

```
import org.blueoxygen.bbook2.DefaultPersistence;

@Entity
@Table(name="event")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
public class Event extends DefaultPersistence {
	private String title;
	private Date date;
	private Set<Person> persons;

	@ManyToMany(cascade={CascadeType.PERSIST, CascadeType.MERGE})
	public Set<Person> getPersons() {
		return persons;
	}
	public void setPersons(Set<Person> person) {
		this.persons = person;
	}
	public Date getDate() {
		return date;
	}
	public void setDate(Date date) {
		this.date = date;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}

}
```

Perhatikan baris pada Event.java, dengan anonasi @ManyToMany

```
	@ManyToMany(cascade={CascadeType.PERSIST, CascadeType.MERGE})
	public Set<Person> getPersons() {
		return persons;
	}
	public void setPersons(Set<Person> person) {
		this.persons = person;
	}
```

Baris diatas menerangkan bahwa isi dari persons adalah berbentuk Java collection, dimana pada kasus ini mengimplementasikan Set. Untuk kasus ini, sebenarnya dapat melakukan implementasi yang bermacam-macam seperti ArrayList, HashMap, ataupun menggunakan implementasi yang lebih advanced menggunakan Jakarta Commons Collections dari Apache.

Bilamana telah selesai, register kedua class tersebut kedalam hibernate.cfg.xml diantara session-factory.

```
<mapping class="org.blueoxygen.bbook2.BusinessPartner"/>
<mapping class="org.blueoxygen.bbook2.Event"/>
```

Jalankan kembali build.xml, untuk membuat table Event dan Person.

Untuk mencoba hubungan antara Event dan Person, dapat dengan menggunakan mekanisme yang sama dengan contoh pertama. Berikut ini adalah sebuah contoh lengkap untuk menambah data kedalam objek Event.

```
package org.blueoxygen.bbook2;

import java.sql.Date;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import org.blueoxygen.bbook2.Event;
import org.blueoxygen.bbook2.Person;
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;


public class HibernateOnetoManyTest {

        public static void main(String[] args) {
                SessionFactory sf;

            Session sess;
            Criteria crit;

            AnnotationConfiguration config = new
                  AnnotationConfiguration();
        config.configure("hibernate.cfg.xml");
        sf = config.buildSessionFactory();
        sess = sf.openSession();
        sess.getTransaction().begin();

        Event event = new Event();
        List<Person> persons = new ArrayList<Person>();
        persons = (ArrayList<Person>)
                sess.createCriteria(Person.class).list();

        event.setTitle("Third Event");
        event.setDate(new Date(System.currentTimeMillis()));
        event.setPersons(new HashSet<Person>());
        for(Person p : persons){
          event.getPersons().add(p);
        }
        sess.save(event);
        sess.getTransaction().commit();

        Person person = (Person) sess.get(Person.class, "1");
        for(Event e : person.getEvents()){
          System.out.println(e.getTitle());
        }
        sf.close();
          }
```

```
}
```

Coba jalankan, pasti error pada baris (Person) sess.get(Person.**class**, "1"), hal ini dikarenakan, method ini mereturn objek Person dengan id=1. Untuk mengatasinya, buatlah sebuah record pada table Person diisi dengan data terlebih dahulu, dengan id = 1. Atau menjalankan build.xml dengan Ant, dengan menjalankan target untuk export-data.

Coba buka table event, maka secara otomatis.

Perhatikan baris untuk mengisi objek Event.

```
        sess.getTransaction().begin();

        Event event = new Event();
        List<Person> persons = new ArrayList<Person>();
        persons = (ArrayList<Person>)
                  sess.createCriteria(Person.class).list();

        event.setTitle("Third Event");
        event.setDate(new Date(System.currentTimeMillis()));
        event.setPersons(new HashSet<Person>());
        for(Person p : persons){
          event.getPersons().add(p);
        }
        sess.save(event);
        sess.getTransaction().commit();
```

Baris ini sebenarnya adalah implementasi JTA (Java Transaction API), yang membuat setiap proses transaksi harus dicommit untuk disetujui untuk direkam kedalam database, bilamana tidak, secara otomatis akan di rollback.

Baris diatas bilamana diartikan adalah sebagai berikut:
- Transaksi dimulai dengan begin()
- Masukan ArrayList dengan isi dari database yaitu person
- Isi variable title dan date pada Event
- Lakukan looping untuk semua isi dari persons (ArrayList)
- Tambahkan setiap user kedalam object Event
- Rekam semuanya dengan perintah save(event)
- Commit semua kegiatan ini.

Jalankan beberapa kali HibernateManyToMany diatas, kemudian cobalah script dibawah ini. Coba buka table event, pilihlah salah satu id, kemudian ganti isian ff88xxx dengan id yang terdapat di table event. Jalankan.

```
Event e = (Event) sess.get(Event.class,
          "ff8080810e205e45010e205e47f00001");
        for(Person p : event.getPersons()){
          System.out.println(p.getFirstname());
}
```

Coba jalankan dengan beberapa id yang berbeda, hasilnya berbeda bukan. Kita tahu table Event tidak memiliki relationship record atau foreign key terhadap Person, dan begitu juga sebaliknya. Tetapi resultnya terfilter berdasarkan idnya bukan. Sebenarnya relationship table

antara event dan person terdapat pada event_person.

Yang hebat dari tipe ManyToMany dari JPA adalah bilamana ada objek bertipe ini, secara otomatis akan membuat table penghubungnya.

Coba ganti-ganti id didalam table baik event atau person, jalankan kembali. Apa yang terjadi? Error akan muncul.

No row with the given identifier exists: [org.blueoxygen.bbook2.Event#ff8080810e205e85010e205e878e0001]

4.9 Implementasi DBUnit sebagai ETL

Hibernate Tools khusus untuk mengenerate table, memiliki beberapa metode, dan secara default adalah menghapus yang ada, kemudian membuatkan yang baru. Dalam implementasinya, metode ini adalah bagus sekali bilamana kita tidak mau pusing dengan table yang digenerate, tetapi ternyata pada implementasinya sering terjadi ada table yang harus diisi terlebih dahulu. Malahan yang lebih tinggi lagi adalah data harus ditransfer dari satu database ke database lain, mengikuti idependensi ORM yang digunakan, seperti yang pernah penulis alami, membuat sebuah implementasi berbasis Hibernate tentu saja, sistem dikembangkan dengan MySQL, client menggantinya menjadi SQL Server. Padahal MySQL sudah dijalankan untuk operasional. Yang lebih parah lagi, pada saat terjadi migrasi, ada beberapa struktur database yang berubah.

Sehingga yang terjadi adalah kita harus melakukan patch terhadap struktur table mengikuti objek Hibernate yang berubah, tetapi isi tetap ada.

Ada dua pendekatan yang dapat dilakukan tentu saja, yaitu menggunakan tools migrasi untuk memindahkan data dari MySQL ke SQL Server, kemudian objek Hibernate yang berubah dimap dengan table-table di SQL Server, dan dirubah satu persatu.

Ternyata ada metode yang lebih baik, yaitu dengan mengimplementasikan ETL singkatan dari Extract Transform Language, sebuah mekanisme yang memungkinkan data di MySQL dipindahkan ke SQL Server atau database manapun bernama DBUnit. Sedangkan HibernateTools tetap diutilisasikan, sehingga Hibernate yang membuat struktur database, dan DBUnit yang mengisi table yang baru terbentuk, dan semua ini hanya dijalankan dengan satu perintah yaitu ant.

Adapun untuk melakukan integrasi migrasi ini dapat dilakukan dengan menggunakan build.xml yang telah kita kembangkan, kemudian ditambahkan dengan task dbunit.

Tentu saja ide ini dapat diimplementasikan bilamana kita mengembangkan produk dengan Hibernate, artinya pihak client dapat menentukan databasenya apa saja, dan Hibernate akan mengenerate table, dan isinya oleh DBUnit, untuk semua database yang didukung Hibernate dan DBUnit tentu saja hal ini terjadi.

Untuk menggunakan DBUnit dalam Ant sebenarnya sangat mudah. Pertama yang paling penting adalah skema untuk export dari database ke format XML.

Scriptnya adalah sebagai berikut

```
<target name="export-data" description="Export data to xml file">
    <taskdef name="dbunit" classpathref="classpath.build"
            classname="org.dbunit.ant.DbUnitTask"/>
```

```
      <dbunit driver="${db.driver}" url="${db.url}" userid="${db.username}"
password="${db.password}">
         <export dest="${db.exportdata}" format="xml"/>
      </dbunit>
      <echo message="Data exported"/>
   </target>
```

Perintah ini berarti mengcopy semua data dari database (${db.url}) menjadi metadata berformat xml. Adapun output dari menjalankan ini dalam Ant adalah

```
Buildfile: /home/frans/workspace/Hibernate/build.xml
export-data:
  [dbunit] Executing export:
  [dbunit] in format: xml to datafile: /home/frans/workspace/Hibernate/dbunt-export.xml
    [echo] Data exported
BUILD SUCCESSFUL
Total time: 2 seconds
```

XML dari kegiatan export ini dapat dikirim ke database lainnya, dan tentu saja harus JDBC compliance. Cara paling mudah adalah dengan menambahkan script berikut pada build.xml

```
        <target name="import-data" description="Add some data to the database">
      <taskdef name="dbunit" classpathref="classpath.build"
            classname="org.dbunit.ant.DbUnitTask"/>
      <dbunit driver="${db.driver}" url="${db.url}" userid="${db.username}"
password="${db.password}">
         <operation type="INSERT" src="${db.testdata}" format="xml"/>
      </dbunit>
      <echo message="Data imported"/>
        </target>
```

Bilamana import-data dijalankan oleh Ant, akan muncul output seperti ini, diluar ini artinya error.

```
Buildfile: /home/frans/workspace/Hibernate/build.xml
import-data:
  [dbunit] Executing operation: INSERT
  [dbunit] on   file: /home/frans/workspace/Hibernate/all-db-data.xml
  [dbunit] with format: xml
    [echo] Data imported
BUILD SUCCESSFUL
Total time: 2 seconds
```

Sedangkan untuk membuat db.driver, db.url, db.username serta db.password dapat dikenal, buatlah sebuah file build.propierties yang diletakan dilokasi yang sama dengan build.xml berada.

DBUnit sebenarnya dapat melakukan kegiatan yang lebih spesifik, seperti membandingkan data dengan isi database tujuan.

4.10 Pengembangan Hibernate Lanjutan

Sebenarnya pengembangan apliksi yang menggunakan ORM akan memiliki keflexisibilitasan terhadap database alias databasenya independent.

Untuk kasus diatas, hanya mengganti jdbc driver dari MySQL ke Oracle, dan mengganti dialect dari MySQLDialect menjadi OracleDialect, secara otomatis aplikasi kita berjalan pada database Oracle.

Kalau diperhatikan lebih lanjut, semua mekansime koneksi, bilamana mapping telah berhasil, adalah melakukan inisialisasi session, dan tentu saja apa yang terjadi bilamana aplikasi lebih kompleks, inisialisasi diinisialisasi disemua objek atau dibuat satu tetapi dishared. Ini merepotkan bukan?

Pada pemograman berikutnya akan dibuat bagaimana mekanisme inisalisasi session baik itu SessionFactory maupun AnnonationConfiguration tidak dilakukan secara manual tetapi otomatis, ini adalah mekanisme yang disebut dengan IOC singkatan dari Injection of Control.