HIBERNATE

# Lý thuyết về Hibernate

## Định nghĩa Hibernate

### Definition

Hibernate is an Object Relational Mapping (ORM) framework for JAVA language. It provides a framework for mapping an object-oriented model to a traditional relational database (MySQL, postgreSQL, SQL Server, …).

Hibernate do JBoss quản lý. Các phiên bản Hibernate gần đây là 3.5, 3.6, 4.0 và 4.1.

Hibernate solves object-relational mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

Hibernate's primary feature is mapping from Java classes to database tables and from Java data types to SQL data types.

Truy vấn dữ liệu bằng: HQL (Hibernate Query Language), SQL truyền thống, API (Application Programming Interface).

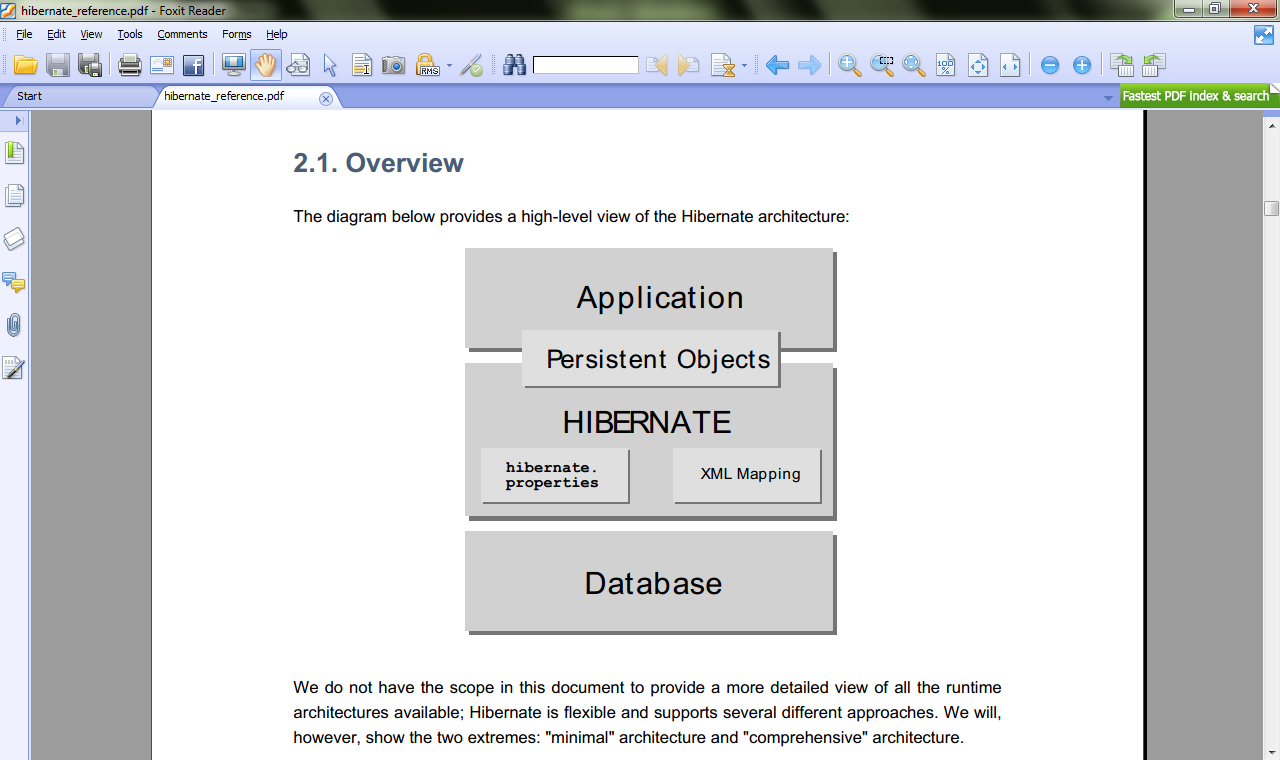


Figure Hibernate architecture

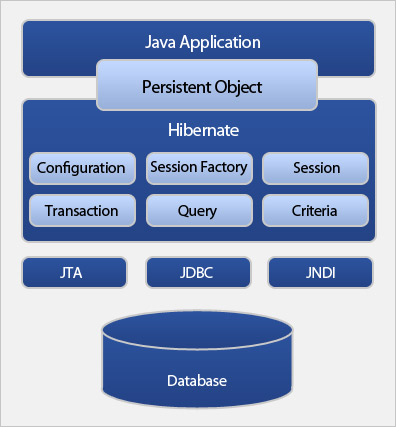


Figure Hibernate architecture

### Cách làm việc của Hibernate

Hibernate uses runtime reflection to determine the persistent properties of a class. The objects to be persisted are defined in a mapping document – XML file, which describes the persistent fields and associations, any subclasses or proxies of the persistent object. The mapping documents are compiled at application startup. They are used in support operations, such as generating the database schema or creating stub Java source files.

A SessionFactory is created from the compiled collection of mapping documents. The SessionFactory provides the mechanism for managing persistent classes, the Session interface. The Session class provides the interface between the persistent data store and the application. The Session interface wraps a JDBC connection and is only used by a single application thread, then closed and discarded.

### Object Relational Mapping (ORM)

Converting data between incompatible type systems in object-oriented programming languages.

This creates a "virtual object database" that can be used from within the programming language. Mapping a data representation from an object model to a relational data model with a SQL-based schema.

Mô tả ORM:

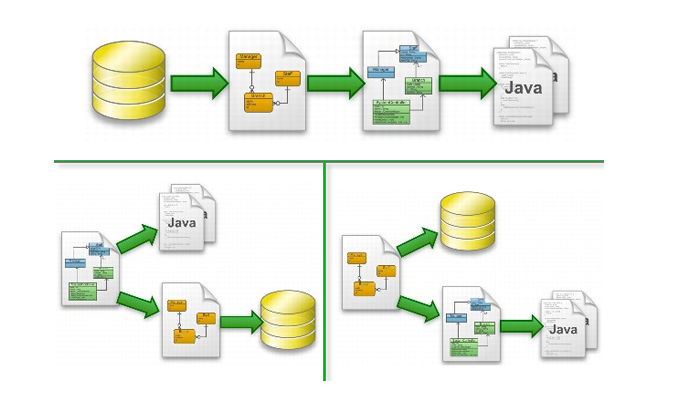


Figure Mô tả ORM

Các dạng mapping:

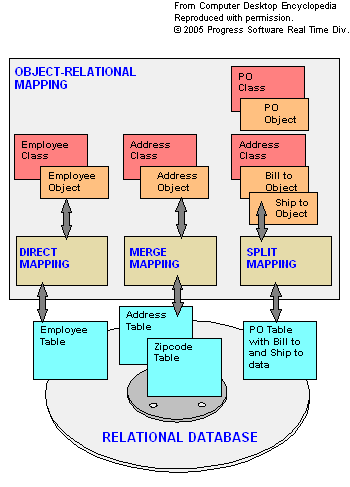


Figure Ba dạng mapping

Hibernate mapping được đặt trong file xml. Tên file thường là \*.hbm.xml. VD về một mapping Stock.hbm.xml:

**<?xml** version="1.0"**?>**

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

**<hibernate-mapping>**

**<class** name="com.mkyong.stock.Stock" table="stock" catalog="mkyongdb"**>**

**<id** name="stockId" type="java.lang.Integer"**>**

**<column** name="STOCK\_ID" **/>**

**<generator** class="identity" **/>**

**</id>**

**<property** name="stockCode" type="string"**>**

**<column** name="STOCK\_CODE" length="10" not-null="true" unique="true" **/>**

**</property>**

**<property** name="stockName" type="string"**>**

**<column** name="STOCK\_NAME" length="20" not-null="true" unique="true" **/>**

**</property>**

**<set** name="stockDailyRecords" table="stock\_daily\_record"

inverse="true" lazy="true" fetch="select"**>**

**<key>**

**<column** name="STOCK\_ID" not-null="true" **/>**

**</key>**

**<one-to-many** class="com.mkyong.stock.StockDailyRecord" **/>**

**</set>**

**</class>**

**</hibernate-mapping>**

Chi tiết các thuộc tính ta tham khảo thêm trong file hibernate-mapping-3.0.dtd.

#### Quan hệ One-to-One

Định nghĩa mapping theo quan hệ One-to-One StockDetail.hbm.xml:

**<?xml** version="1.0"**?>**

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

*<!-- Generated 25 April 2011 7:52:33 PM by Hibernate Tools 3.4.0.CR1 -->*

**<hibernate-mapping>**

**<class** name="com.mkyong.stock.StockDetail" table="stock\_detail"

catalog="mkyongdb"**>**

**<id** name="stockId" type="java.lang.Integer"**>**

**<column** name="STOCK\_ID" **/>**

**<generator** class="foreign"**>**

**<param** name="property"**>**stock**</param>**

**</generator>**

**</id>**

**<one-to-one** name="stock" class="com.mkyong.stock.Stock"

constrained="true"**></one-to-one>**

**<property** name="compName" type="string"**>**

**<column** name="COMP\_NAME" length="100" not-null="true" **/>**

**</property>**

**<property** name="compDesc" type="string"**>**

**<column** name="COMP\_DESC" not-null="true" **/>**

**</property>**

**<property** name="remark" type="string"**>**

**<column** name="REMARK" not-null="true" **/>**

**</property>**

**<property** name="listedDate" type="date"**>**

**<column** name="LISTED\_DATE" length="10" not-null="true" **/>**

**</property>**

**</class>**

**</hibernate-mapping>**

#### Quan hệ One-to-Many

Định nghĩa StockDailyRecord.hbm.xml theo quan hệ One-to-Many

**<?xml** version="1.0"**?>**

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

**<hibernate-mapping>**

**<class** name="com.mkyong.stock.StockDailyRecord" table="stock\_daily\_record"

catalog="mkyongdb"**>**

**<id** name="recordId" type="java.lang.Integer"**>**

**<column** name="RECORD\_ID" **/>**

**<generator** class="identity" **/>**

**</id>**

**<many-to-one** name="stock" class="com.mkyong.stock.Stock" fetch="select"**>**

**<column** name="STOCK\_ID" not-null="true" **/>**

**</many-to-one>**

**<property** name="priceOpen" type="java.lang.Float"**>**

**<column** name="PRICE\_OPEN" precision="6" **/>**

**</property>**

**<property** name="priceClose" type="java.lang.Float"**>**

**<column** name="PRICE\_CLOSE" precision="6" **/>**

**</property>**

**<property** name="priceChange" type="java.lang.Float"**>**

**<column** name="PRICE\_CHANGE" precision="6" **/>**

**</property>**

**<property** name="volume" type="java.lang.Long"**>**

**<column** name="VOLUME" **/>**

**</property>**

**<property** name="date" type="date"**>**

**<column** name="DATE" length="10" not-null="true" unique="true" **/>**

**</property>**

**</class>**

**</hibernate-mapping>**

#### Quan hệ Many-to-Many

Định nghĩa file Category.hbm.xml có quan hệ Many-to-Many

**<?xml** version="1.0"**?>**

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

**<hibernate-mapping>**

**<class** name="com.mkyong.stock.Category" table="category" catalog="mkyongdb"**>**

**<id** name="categoryId" type="java.lang.Integer"**>**

**<column** name="CATEGORY\_ID" **/>**

**<generator** class="identity" **/>**

**</id>**

**<property** name="name" type="string"**>**

**<column** name="NAME" length="10" not-null="true" **/>**

**</property>**

**<property** name="desc" type="string"**>**

**<column** name="[DESC]" not-null="true" **/>**

**</property>**

**<set** name="stocks" table="stock\_category" inverse="true" lazy="true" fetch="select"**>**

**<key>**

**<column** name="CATEGORY\_ID" not-null="true" **/>**

**</key>**

**<many-to-many** entity-name="com.mkyong.stock.Stock"**>**

**<column** name="STOCK\_ID" not-null="true" **/>**

**</many-to-many>**

**</set>**

**</class>**

**</hibernate-mapping>**

### Hibernate Configuration File

Thường đặt tên hibernate.cfg.xml. Ví dụ:

**<?xml** version="1.0" encoding="utf-8"**?>**

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

**<hibernate-configuration>**

**<session-factory>**

**<property** name="hibernate.connection.driver\_class"**>**com.mysql.jdbc.Driver**</property>**

**<property** name="hibernate.connection.url"**>**jdbc:mysql://localhost:3306/mkyongdb**</property>**

**<property** name="hibernate.connection.username"**>**root**</property>**

**<property** name="hibernate.connection.password"**>**password**</property>**

**<property** name="hibernate.dialect"**>**org.hibernate.dialect.MySQLDialect**</property>**

**<property** name="show\_sql"**>**true**</property>**

**<mapping** resource="com/mkyong/stock/Stock.hbm.xml" **/>**

**<mapping** resource="com/mkyong/stock/StockDetail.hbm.xml" **/>**

**</session-factory>**

**</hibernate-configuration>**

Chi tiết từng thuộc tính tham khảo trong file hibernate-configuration-3.0.dtd.

## Persistence layer

Mô hình 3 lớp:

* Presentation layer
* Business layer
* Data layer

Trong đó, business layer có thể chia ra thành 2 lớp nhỏ:

* Business logic layer : các tính toán nhằm thỏa mãn yêu cầu người dùng.
* Persistence layer : chịu trách nhiệm giao tiếp với data layer.

Persistence sẽ đảm nhiệm các nhiệm vụ mở kết nối, truy xuất và lưu trữ dữ liệu vào các RDBMS (Relational DataBase Management System).

Hình mô tả vị trí của tầng persistence trong kiến trúc n-tier:

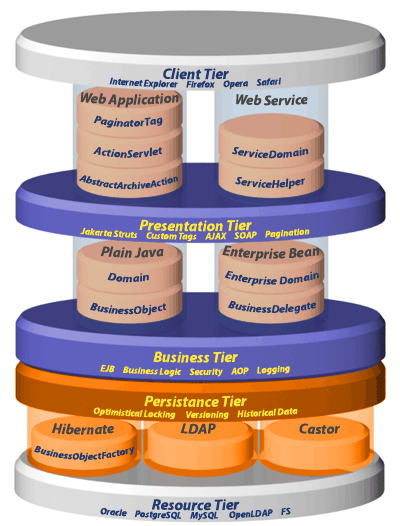


Figure Vị trí của tầng persistence

## HQL (Hibernate Query Language)

Khác biệt căn bản giữa HQL và SQL là: dùng lớp thay cho bảng, property thay cho cột.

### Truy vấn dữ liệu

Retrieve a stock data where stock code is “7277″.

Query query = session.createQuery("from Stock where stockCode = :code ");

query.setParameter("code", "7277");

List list = query.list();

hoặc:

Query query = session.createQuery("from Stock where stockCode = '7277' ");

List list = query.list();

### Cập nhật dữ liệu

Update a stock name to “DIALOG1″ where stock code is “7277″.

Query query = session.createQuery("update Stock set stockName = :stockName" +

" where stockCode = :stockCode");

query.setParameter("stockName", "DIALOG1");

query.setParameter("stockCode", "7277");

**int** result = query.executeUpdate();

hoặc

Query query = session.createQuery("update Stock set stockName = 'DIALOG2'" +

" where stockCode = '7277'");

**int** result = query.executeUpdate();

### Xóa dữ liệu

Delete a stock where stock code is “7277″.

Query query = session.createQuery("delete Stock where stockCode = :stockCode");

query.setParameter("stockCode", "7277");

**int** result = query.executeUpdate();

hoặc

Query query = session.createQuery("delete Stock where stockCode = '7277'");

**int** result = query.executeUpdate();

### Thêm dữ liệu

In HQL, only the INSERT INTO … SELECT … is supported; there is no INSERT INTO … VALUES. HQL only support insert from another table. For example

"insert into Object (id, name) select oo.id, oo.name from OtherObject oo";

Insert a stock record from another backup\_stock table. This can also called bulk-insert statement.

Query query = session.createQuery("insert into Stock(stock\_code, stock\_name)" +

"select stock\_code, stock\_name from backup\_stock");

**int** result = query.executeUpdate();

The query.executeUpdate() will return how many number of record has been inserted, updated or deleted.

# Sử dụng Hibernate

## Mapping Java class đơn giản vào Hibernate

### Đặt vấn đề

Ví dụ, ta có class đơn giản (JavaBean class) cần mapping vào Hibernate.

**public** **class** Event {

**private** **long** id;

**private** String title;

**private** Date date;

**public** Event() {

}

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** id) {

**this**.id = id;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

}

Trong đó, id đóng vai trò như khóa, date và title là các thuộc tính.

### Thực hiện mapping

Mapping class trên và Hibernate (sử dụng file XML)

<?xml version=*"1.0"*?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package=*"org.hibernate.tutorial.domain"*>

<class name=*"Event"* table=*"EVENTS"*>

<id column=*"EVENT\_ID"* name=*"id"* type=*"long"*>

<generator class=*"assigned"* />

</id>

<property column=*"EVENT\_DATE"* generated=*"never"* lazy=*"false"*

name=*"date"* type=*"timestamp"* />

<property generated=*"never"* lazy=*"false"* name=*"title"* type=*"string"* />

</class>

</hibernate-mapping>

Giải thích:

* org.hibernate.tutorial.domain là package của class Event.
* Event là tên class Java.
* EVENTS là tên bảng trong database.
* Thẻ <id> dùng định nghĩa tương đương khóa chính duy nhất trong bảng. Cột tương ứng trong bảng là EVENT\_ID.
* Thẻ <property> dùng mapping các field thông thường của bảng.
* Các tên date và title là tên thuộc tính của lớp Event.
* EVENT\_DATE là tên cột trong bảng EVENTS tương ứng với thuộc tính date.
* Thuộc tính title không chỉ định column nên trong database tên cột cũng là title.
* Lưu ý: type ở đây không phải là của database hay java, nó là Hibernate mapping type.

Khi khai báo như trên, Hibernate sẽ phát sinh các hàm để get, set dữ liệu tương ứng: getId(), setId(), getDate(), setDate(), getTitle(), setTitle().

Hibernate supports for two third-party open source JDBC connection pools: c3p0 and proxool.

## Xây dựng Liferay hibernate project demo

Mục tiêu của project: xây dựng project tương tác cơ sở dữ liệu thông qua Hibernate mà không sử dụng service builder của Liferay IDE.

### Tạo project

Dùng Liferay 6.1.0. Tạo một project MVC qlth (quản lý trường học). Tạo các đối tượng: Sinh Viên, Lớp Học, Môn Học, Kết Quả.

Mối quan hệ giữa các đối tượng này như sau:

* Lớp Học và Sinh Viên có quan hệ một - nhiều.
* Sinh Viên và Môn Học có quan hệ nhiều - nhiều thông qua Kết Quả.

Khi tạo project qlth thì không tạo service builder. Sử dụng database demo bằng MySQL.

Đặt project trong Demo-6.1.0, tạo 4 bảng trong CSDL mySQL: qlth\_ketqua, qlth\_lophoc, qlth\_monhoc, qlth\_sinhvien. Thêm 3 portlet: QL LopHoc, QL MonHoc, QL SinhVien.

Xóa portlet đi kèm theo project khi tạo mới (xóa trong các file xml của WEB-INF và html).

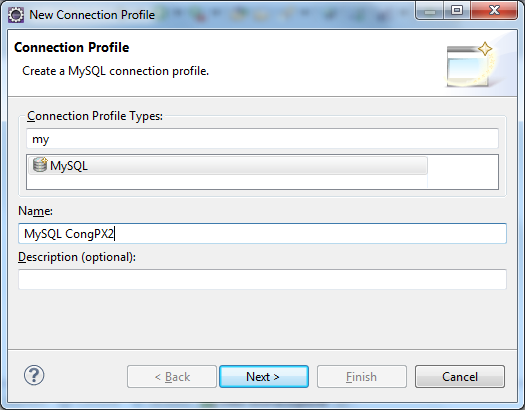
### Từng bước xây dựng

Download và cài đặt JBoss Tools cho Eclipse Juno (có thể thông qua Eclipse Marketplace).

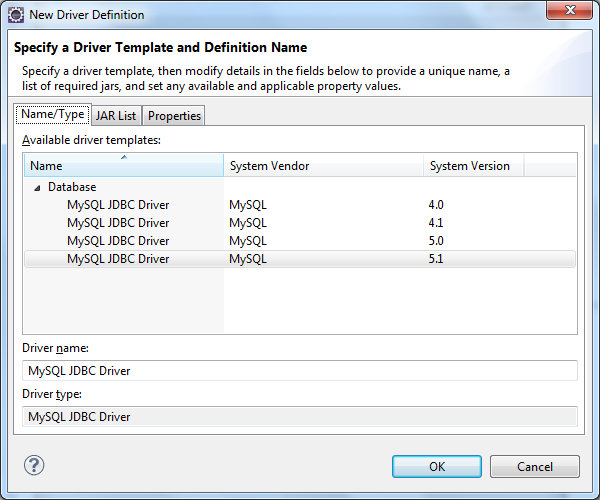
#### Data Source Explorer

Trong Eclipse, chọn Window 🡺 Show View 🡺 Data Source Explorer.

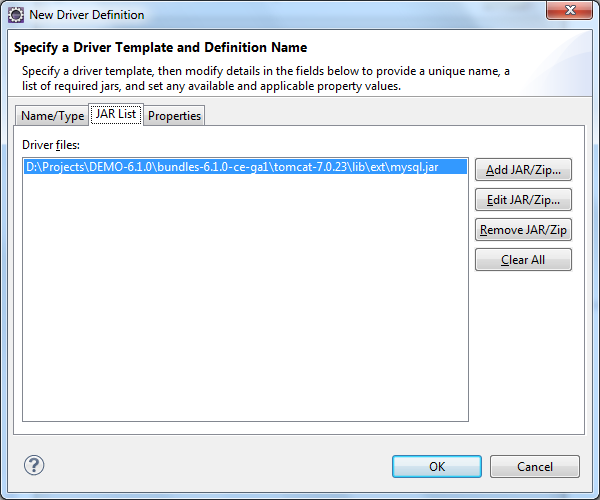
Phải chuột vào Database Connections 🡺 New. Chọn kiểu kết nối là MySQL.



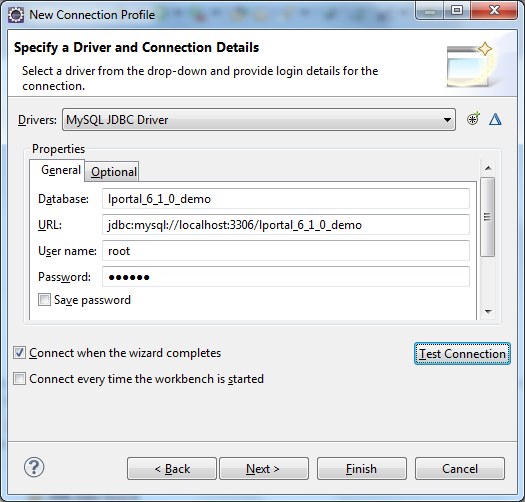
Chọn phiên bản MySQL đang kết nối đến.



Chỉ định gói jar kết nối MySQL.

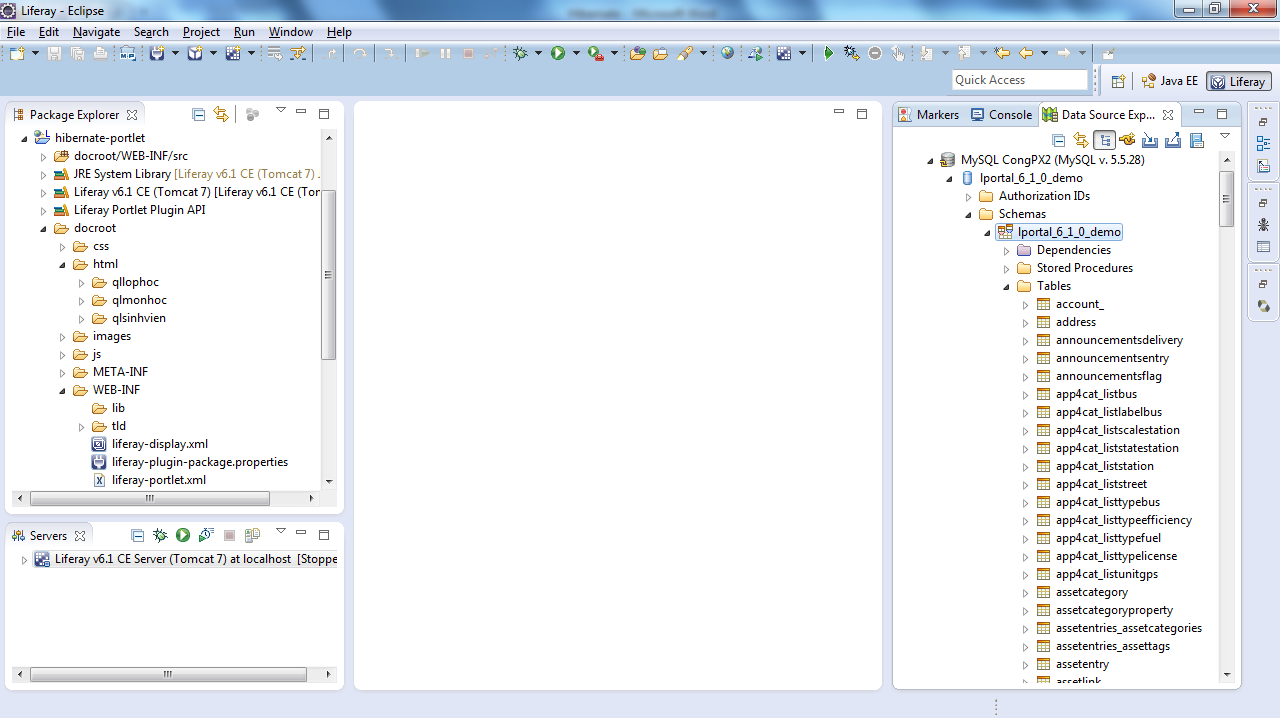


Kiểm tra và hoàn tất kết nối.



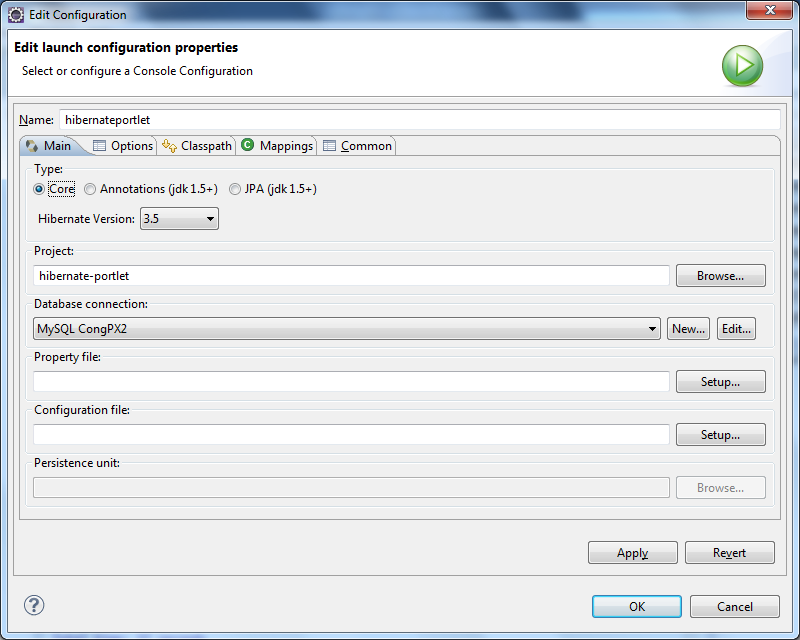
Kiểm tra kết nối

Xem đối tượng trong kết nối như hình dưới.

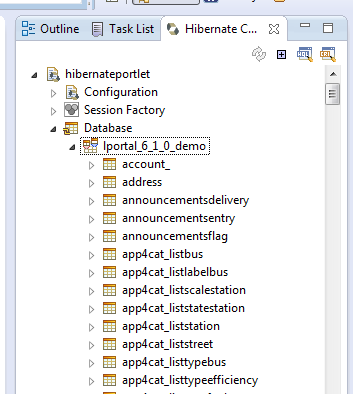


#### Hibernate Configuration

Mở cửa sổ Hibernate Configurations, chọn Add Configuration... Nhập thông tin cấu hình như hình dưới:



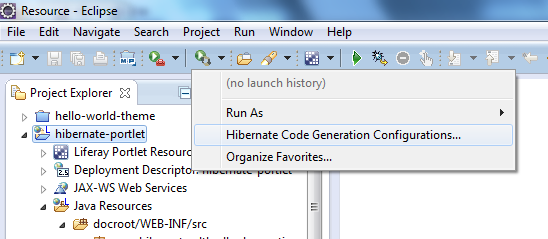
Kiểm tra lại kết quả:



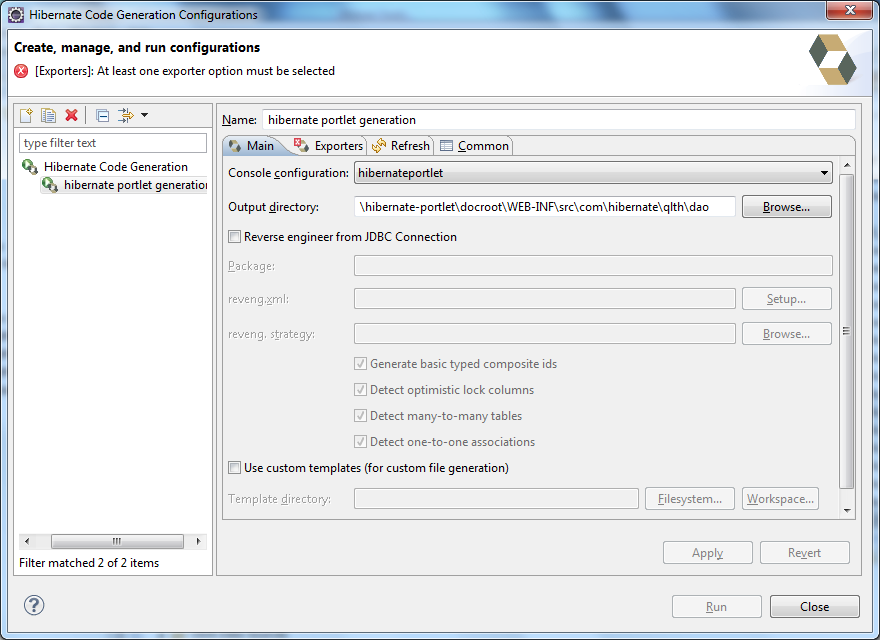
#### Cấu hình Hibernate Code Generation

Tạo package trong source: com.hibernate.qlth.dao.

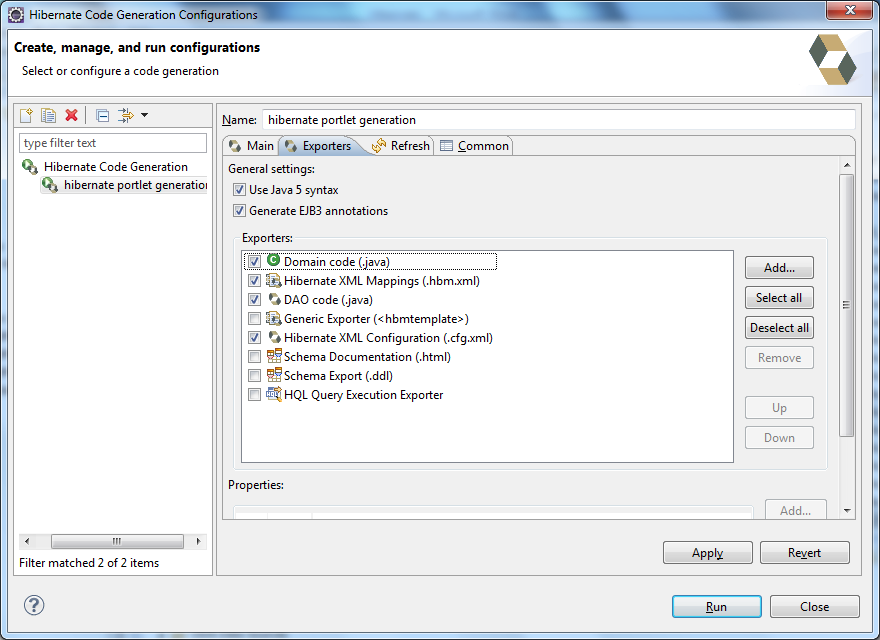
Chọn configuration từ biểu tượng:



Nhập thông tin cấu hình:



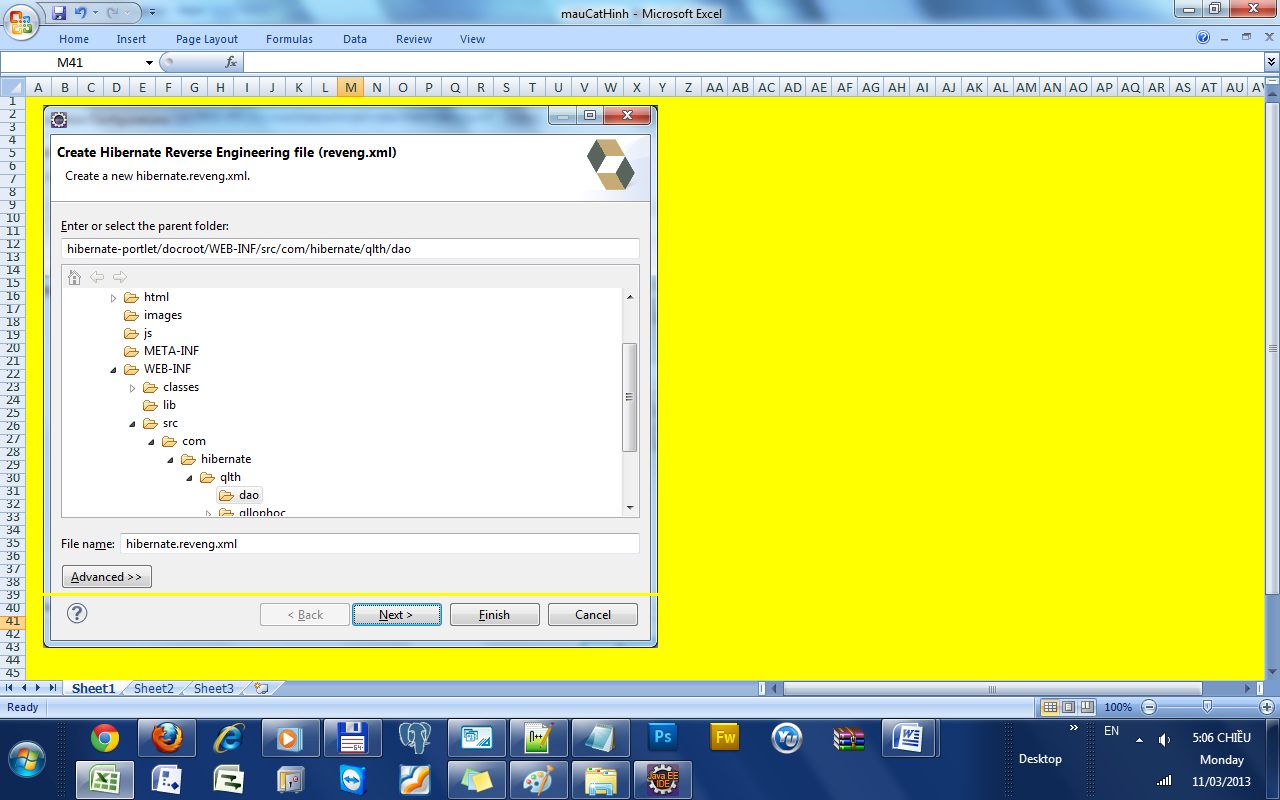
Tab Exporters:



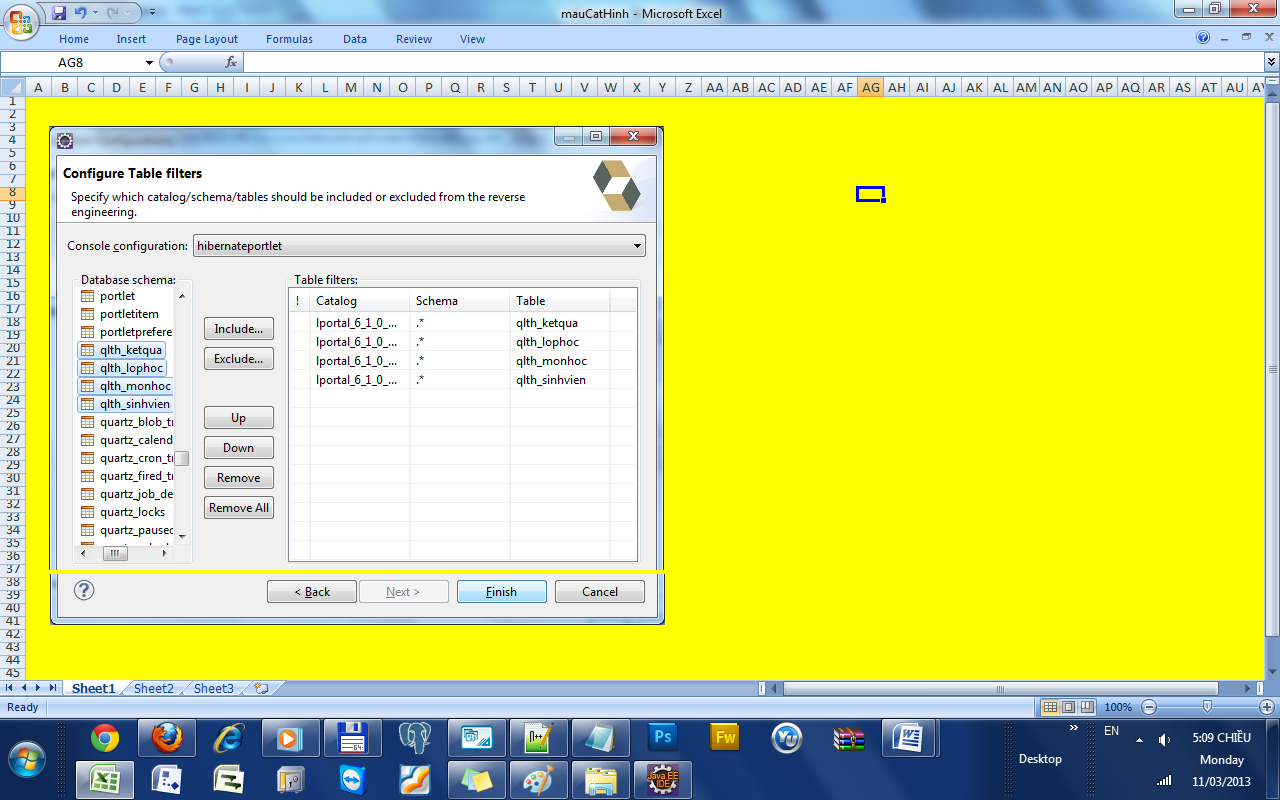
Nhấn Apply và nhấn Run.

**Phần phát sinh hbm và code thông qua reverse**

Hiệu chỉnh configuration bên trên. Check vào "Reverse engineer from JDBC Connection". Trong reveng.xml click vào Setup 🡺 Create New.

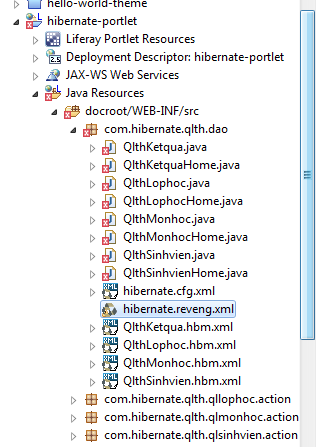


Chọn bảng để phát sinh mã lệnh:



Sau đó, nhấn Run để phát sinh code.

Kiểm tra lại kết quả:



Ta dựa vào các file phát sinh ra để thao tác dữ liệu.

fdsa

gj

fdsfakl fsfda

fdsfla

ad

ffas