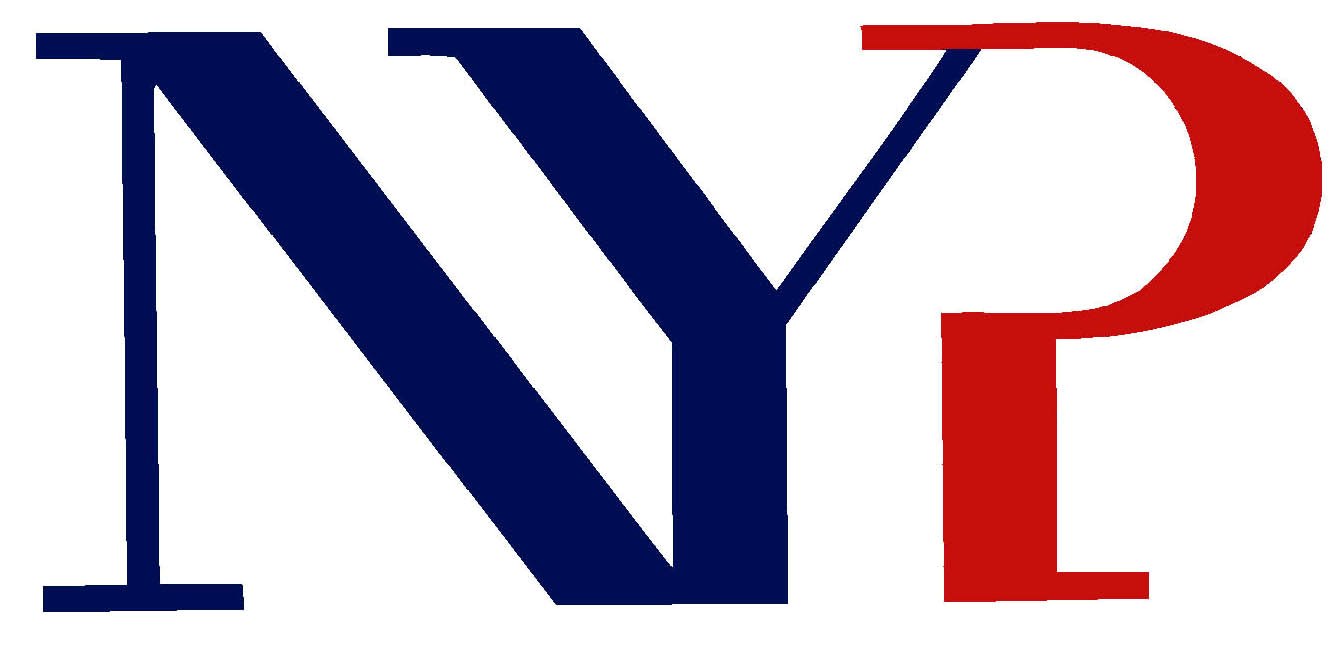
****

**IT2274 Java Enterprise Development Assignment**

**Research on Hibernate ORM**

WONG KANG FEI

124465R

Class IT1206

(65) 98739041

[124465R@mymail.nyp.edu.sg](mailto:124465R@mymail.nyp.edu.sg)

Diploma in Information Technology

Written Assignment for IT2274 Java Enterprise Development

Mr Law Chee Yong

2013 Semester 2

ICA

**Introduction**

In our ideal world, it would be trivial to take any Java object and persist it to the database, except that when using JDBC requires a considerable body of code, it would really be troublesome when it comes to writing the same SQL over and over again. What if there's SQL syntax error? Or sometimes mysterious errors because of misconfiguration? Not to mention we might have over hundreds or even thousands of Java object we need to persist to the database. How should we address this problem we face in every project?

**Hibernate As a Persistence Solution**

Hibernate is a open source project currently lead by RedHat. It alleviates some of the pain we usually face such as reducing the boilerplate code for opening connections and logging problem. It allows the developers focus more on the business logic and increase productivity. Hibernate also reduce the lines of code needed, making the system more understandable and easier to refactor. It abstracts our application away from the underlying SQL database, which means that switching to other SQL database requires few changes in Hibernate configuration file.

**Why Hibernate?**

There are several similar technologies that address the problem as Hibernate trying to solve. Hibernate is built on top of JPA with extra specifications that is unique to Hibernate. However, personally I would choose Hibernate over other technologies based on the following points:

* Learning curve is shorter compare to other framework
* Has code generation tool
* Has optional modules that work seamlessly with Hibernate (more on this later)
* Demands in market
* It's free
* Popular

**Environtment setup**

Before we get to use Hibernate in our application, there are a few configuration we need to make sure.

* Download JDBC Connector/J driver from <http://dev.mysql.com/downloads/connector/j/>.
* Extract and include JDBC Connector/J driver to the application classpath.
* Download Hibernate library from [www.hibernate.org](http://www.hibernate.org/). It's recommended to use Maven to handle dependency issue, but since I have little experience with Maven, I choose to download the whole package from <http://sourceforge.net/projects/hibernate/files/hibernate4/>.
* Extract and include all the .jar files under “lib/required” and “lib/provided” folder to the application classpath.
* Create a Hibernate XML configuration file (hibernate.cfg.xml) in “src” folder that points the database and mapped classes.

I've included a project “it2274-hibernate-hello-world” to demonstrate the minimal working exmaple. Please refer to my comments in the classes and configuration file for furthur explaination.

**Mapping Java Classes**

In order for Hibernate to know which classes to be presisted, we need to map the classes.

A clear understanding in UML is needed, as we deal with a lot of mapping such as One-to-Many, Many-to-Many relationship and etc.

There are two ways to map a Java class, by Annotation or XML. These two approach has their own pros ans cons. In brief, my own opinion is that to use Annotation for readability and XML for portability. After that, a declaration need to be added to the hibernate.cfg.xml file. In the examples I will use Annotation to map the classes.

**Schema Generation**

It's not rnecessary to create a schema manually. Hibernate is able to generate the tables and properties needed in the schema base on the mapped classes. That is, simply provide the necessary configuration to an empty schema in the hibernate.cfg.xml and declaration of the mapped classes. Changing the attributes of the mapped classes will also trigger changes in schema accordingly, which is very useful during development phase.

**Hibernate Optional Modules**

There are several optional modules provide out-of-the-box feature on top of Hibernate. I've been reasearching two of the available modules and has successfully implemented them into my semestral project.

Hibernate Envers is an auditing module that works similar to the concept of revision control such as SVN and GIT. It makes auditing easy such that every transaction to the database are revisioned, including the changes of the values and timestamp, even if the record is deleted. You can query for entities changed at specific revision or vice versa.

Hibernate Search is a full text search library build on top of Apache Lucene. It is capable of performing indexing on the database to optimize the search performance. More importantly, it provides search API that is able to handle complicated SQL statement to the database. It also make use of Solr Framework for analyzing and filtering to achieve more inteligent search results.

**Hibernate Hello World Example**

This example demonstrate the minimal working application using Hibernate. You'll probably need to change the configuration I've made in the hibernate.cfg.xml. The only part you have to edit is the JDBC connection parameters.

<!-- Standard JDBC configuration parameters -->

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

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/it2274\_hibernate\_example</property>

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

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

You must also make sure that you have created an empty schema call it2274\_hibernate\_example in your MySQL instance.

For code explanation, refer to my comments in the source files.