# Hibernate General Questions

* Wha is hibernate?

Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database

Hibernate facilitates the storage and retrieval of Java domain objects via Object/Relational Mapping.

http://javabeginnerstutorial.com/hibernate/hibernate-framework-basic/

* What Hibernate Does?
* Map Java class to database tables & vice versa
* Data query and retrieval facility
* Generate the SQL query based on the underline DB. and attempts to relieve the developer from manual result set handling and object conversion.
* Make application portable to all relational DB.
* Enhance performance by providing the different levels of cache(First, Second and Query level).

http://javabeginnerstutorial.com/hibernate/hibernate-framework-basic/

* What is HQL?

HQL is abbreviation of **Hibernate Query Language.** It is SQL inspired language provided by hibernate. Developer can write SQL like queries to work with data objects.

http://javabeginnerstutorial.com/hibernate/hibernate-framework-basic/

* What is Dialect?

Each SQL vendor has its own set of supported syntax. This is known as dialect. In order to generate appropriate sql query Hibernate needs to know, for which DB query needs to be generated. Hibernate does it by org.hibernate.dialect.Dialect class and its subclass for each vendor.

E.g.:

DB2       :   org.hibernate.dialect.DB2Dialect

MySQL  :   org.hibernate.dialect.MySQLDialects

* What is Connection Pooling?

Connection pooling is a technique to open/prepare/close connections. A connection pooling mechanism is a piece of software (component), to which you delegate the function of managing connections. Your application would just ask for a connection, use it, and deliver it back to the pool. The component is responsible for opening N connections and leave them ready for when your application asks. If a connection is stale, the pooling mechanism would then close it and reopen a new one. This represents a better usage of connections, as you don't need to wait for the connection to be established during the actual execution of your code and you don't have to worry about stale connections.

Hibernate doesn't really ship any real connection pooling mechanism. It provides an internal connection manager, which is very rudimentary. The reason is simple: almost (if not all) Application Servers (like JBoss AS) and Servlet Containers (like Tomcat) provides a connection pooling mechanism by default. Thus, your application don't have to worry about the details about it. It just asks the AS for a connection.

In my opinion, there are only two cases where you need to worry about connection pooling:

1. You are dealing with a standalone application (which doesn't run inside a container)
2. You are really expert in connection pooling and none of the existing suits your needs.

But in my experience, most people that uses an "external" connection pooling do so for lack of knowledge about connection pooling *and* lack of knowledge about their container.

Third party connection pooling for production use.

* c3p0 connection pool
* Proxool connection pool
* Obtaining connections from an application server, using JNDI
* Example of configuration.cfg file.

<hibernate-configuration>

<session-factory>

<!-- Database connection settings -->

<property name="**connection.driver\_class**">org.h2.Driver</property>

<property name="**connection.url**">jdbc:h2:mem:db1;DB\_CLOSE\_DELAY=-1;MVCC=TRUE</property>

<property name="**connection.username**">sa</property>

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

<!-- JDBC connection pool (use the built-in) -->

<property name="**connection.pool\_size**">1</property>

<!-- SQL dialect -->

<property name="**dialect**">org.hibernate.dialect.H2Dialect</property>

<!-- Disable the second-level cache -->

<property name="cache.provider\_class">org.hibernate.cache.internal.NoCacheProvider</property>

<!-- Echo all executed SQL to stdout -->

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

<!-- Drop and re-create the database schema on startup -->

<property name="**hbm2ddl.auto**">create</property>

<!-- The mapping information of entities -->

<mapping class="hibernate\_example.Book"/>

</session-factory>

* Give spring architecture.
* Give spring architecture.
* Give spring architecture.
* Give spring architecture.
* Give spring architecture.
* Give spring architecture.
* Give spring architecture.
* Give spring architecture.
* Give spring architecture.
* Give spring architecture.