**Difference between get and load in Hibernate?**

**get will hit the database** if object is not found in the cache and returned completely initialized object, which may involve several database call while **load() method can return proxy**, if object is not found in cache and only hit database if any method other than getId() is called. This can save lot of performance in some cases

**Difference between save, persist and saveOrUpdate methods in Hibernate?**

save(), saveOrUpdate() and persist() is used to save objects into database, but has subtle differences e.g. save() can only INSERT records but saveOrUpdate() can either [INSERT or UPDATE records](http://javarevisited.blogspot.com/2013/01/jdbc-batch-insert-and-update-example-java-prepared-statement.html). Also, return type of save() is a Serializable object, while return type of persist() method is void. You can also check [save vs persist vs saveOrUpdate](http://javarevisited.blogspot.com/2012/09/difference-hibernate-save-vs-persist-and-saveOrUpdate.html) for complete differences between them in hibernate.

Difference between save and persist method in Hibernate

In last section we saw What are difference between save and saveOrUpdate and now we will see Difference on save vs persist method.

1)First difference between save and persist is there return type. Similar to save method persist also INSERT records into database but **return type of persist is void** while return type of save is [Serializable](http://javarevisited.blogspot.sg/2012/01/serializable-externalizable-in-java.html) object.

2) Another difference between persist and save is that both methods make a [transient](http://javarevisited.blogspot.sg/2012/03/difference-between-transient-and.html) instance persistent. However, persist() method doesn't guarantee that the identifier value will be assigned to the persistent instance immediately, the assignment might happen at flush time.

3) One more thing which differentiate persist and save method in Hibernate is that is there behavior on outside of transaction boundaries. persist() method guarantees that it will not execute an INSERT statement if it is called outside of [transaction boundaries](http://javarevisited.blogspot.sg/2011/11/database-transaction-tutorial-example.html). save() method does not guarantee the same, it returns an identifier, and if an INSERT has to be executed to get the identifier (e.g. "identity" generator), this INSERT happens immediately, no matter if you are inside or outside of a transaction.

**What is named SQL query in Hibernate?**

This Hibernate Interview question is related to query functionality provided by Hibernate. Named queries are SQL queries which are defined in mapping document using <sql-query> tag and called using Session.getNamedQuery() method. Named query allows you to refer a particular query by the name you provided, by the way you can define named query in hibernate either by using annotations or xml mapping file, as I said above. @NameQuery is used to define single named query and @NameQueries is used to define multiple named query in hibernate.

**What is SessionFactory in Hibernate? is SessionFactory thread-safe?**

SessionFactory is often built during start-up and used by application code to get session object. It acts as single data store and its also [**thread-safe**](http://javarevisited.blogspot.com/2012/12/how-to-create-thread-safe-singleton-in-java-example.html) so that multiple thread can use same SessionFactory. Usually a Java JEE application has just one SessionFactory, and individual threads, which are servicing client’s request obtain hibernate Session instances from this factory, that’s why any implementation of SessionFactory interface must be thread-safe. Also internal state of SessionFactory, which contains all meta data about Object/Relational mapping is [Immutable](http://javarevisited.blogspot.com/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html) and can not be changed once created.

**What is Session in Hibernate? Can we share single Session among multiple threads in Hibernate?**

Session represent a small unit of work in Hibernate, they maintain connection with database and they are **not thread-safe**.

**What is difference between sorted and ordered collection in hibernate?**

This is one of the easy Hibernate interview question you ever face. A sorted collection is sorted in memory by using [Java Comparator](http://java67.blogspot.com/2012/10/how-to-sort-object-in-java-comparator-comparable-example.html), while a ordered collection uses database's order by clause for ordering. For large data set it's better to use ordered collection to avoid any [OutOfMemoryError in Java](http://javarevisited.blogspot.com/2011/09/javalangoutofmemoryerror-permgen-space.html), by trying to sort them in memory.

**What is difference between transient, persistent and detached object in Hibernate?**

In Hibernate, Object can remain in three state transient, persistent or detached. An object which is associated with Hibernate session is called persistent object. Any change in this object will reflect in database based upon your flush strategy i.e. automatic flush whenever any property of object change or explicit flushing by calling Session.flush() method. On the other hand if an object which is earlier associated with Session, but currently not associated with it are called detached object. You can reattach detached object to any other session by calling either update() or saveOrUpdate() method on that session. Transient objects are newly created instance of persistence class, which is never associated with any Hibernate Session. Similarly you can call persist() or save() methods to make transient object persistent. Just remember, here transient doesn’t represent [transient keyword in Java](http://javarevisited.blogspot.com/2012/03/difference-between-transient-and.html), which is altogether different thing.

**What does Session lock() method do in Hibernate?**

This one is one of the tricky Hibernate Interview question, because Session's lock() method **reattach object without synchronizing or updating with database**. So you need to be very careful while using lock() method. By the way you can always use Session's update() method to sync with database during reattachment. Some time this Hibernate question is also asked as *what is difference between Session's lock() and update() method*. You can use this key point to answer that question as well.

**What is Second level Cache in Hibernate?**

This is one of the first interview question related to caching in Hibernate, you can expect few more. Second level Cache is maintained at SessionFactory level and can improve performance by saving few [database round trip](http://javarevisited.blogspot.com/2012/01/improve-performance-java-database.html). Another worth noting point is that second level cache is available to whole application rather than any particular session.

**What is query cache in Hibernate ?**

QueryCache actually stores result of sql query for future calls. Query cache can be used along with second level cache for improved performance. Hibernate support various open source caching solution to implement Query cache e.g. EhCache.

**Why it's important to provide no argument constructor in Hibernate Entities?**  
Every Hibernate Entity class must contain a [no argument constructor](http://javarevisited.blogspot.com/2012/12/what-is-constructor-in-java-example-chainning-overloading.html), because Hibernate framework creates instance of them using Reflection API, by calling Class.newInstance() method. This method will throw InstantiationException if it doesn't found no argument constructor inside Entity class.  
  
**Can we make an Hibernate Entity Class final?**  
Yes, you can make an Hibernate Entity class final, but that's not a good practice. Since Hibernate uses proxy pattern for performance improvement in case of lazy association, by making an entity final, Hibernate will no longer be able to use proxy, because [Java doesn't allow extension of final class](http://javarevisited.blogspot.com/2011/12/final-variable-method-class-java.html), thus limiting your performance improvement options. Though, you can avoid this penalty, if your persistent class is an implementation of interface, which declares all public methods defined in Entity class.

**1.What is ORM ?**

ORM stands for object/relational mapping. ORM is the automated persistence of objects in a Java application to the tables in a relational database.

|  |
| --- |
|  |

**2.What does ORM consists of ?**

An ORM solution consists of the followig four pieces:

* API for performing basic CRUD operations
* API to express queries refering to classes
* Facilities to specify metadata
* Optimization facilities : dirty checking,lazy associations fetching

**3.What are the ORM levels ?**

The ORM levels are:

* Pure relational (stored procedure.)
* Light objects mapping (JDBC)
* Medium object mapping
* Full object Mapping (composition,inheritance, polymorphism, persistence by reachability)

**4.What is Hibernate?**

Hibernate is a pure Java object-relational mapping (ORM) and persistence framework that allows you to map plain old Java objects to relational database tables using (XML) configuration files.Its purpose is to relieve the developer from a significant amount of relational data persistence-related programming tasks.

|  |
| --- |
|  |

**5.Why do you need ORM tools like hibernate?**

The main advantage of ORM like hibernate is that it shields developers from messy SQL. Apart from this, ORM provides following benefits:

* **Improved productivity**
  + High-level object-oriented API
  + Less Java code to write
  + No SQL to write
* **Improved performance**
  + Sophisticated caching
  + Lazy loading
  + Eager loading
* **Improved maintainability**
  + A lot less code to write
* **Improved portability**
  + ORM framework generates database-specific SQL for you

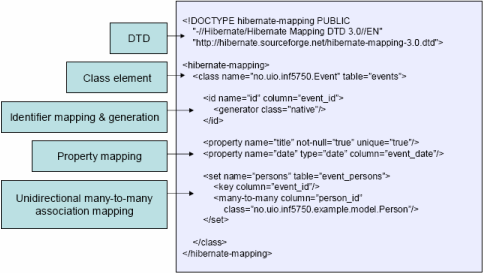
**6.What Does Hibernate Simplify?**

Hibernate simplifies:

* Saving and retrieving your domain objects
* Making database column and table name changes
* Centralizing pre save and post retrieve logic
* Complex joins for retrieving related items
* Schema creation from object model

**7.What is the need for Hibernate xml mapping file?**

Hibernate mapping file tells Hibernate which tables and columns to use to load and store objects. Typical mapping file look as follows:

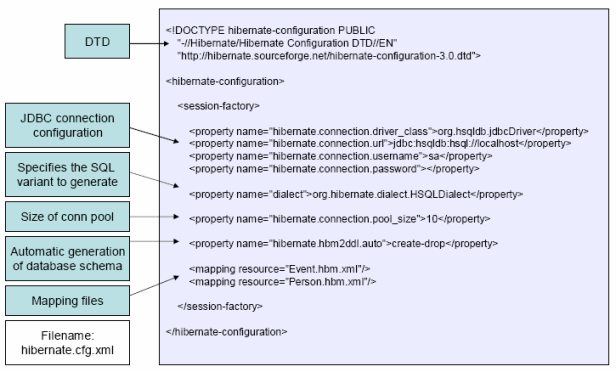
  
  
**8.What are the most common methods of Hibernate configuration**?

The most common methods of Hibernate configuration are:

* Programmatic configuration
* XML configuration (hibernate.cfg.xml)

**9.What are the important tags of hibernate.cfg.xml?**

Following are the important tags of hibernate.cfg.xml:

  
 **10.What are the Core interfaces are of Hibernate framework?**

|  |
| --- |
|  |

The five core interfaces are used in just about every Hibernate application. Using these interfaces, you can store and retrieve persistent objects and control transactions.

* Configuration interface
* Session interface
* SessionFactory interface
* Transaction interface
* Query and Criteria interfaces

|  |
| --- |
|  |

**11.What role does the Session interface play in Hibernate?**

The Session interface is the primary interface used by Hibernate applications. It is a single-threaded, short-lived object representing a conversation between the application and the persistent store. It allows you to create query objects to retrieve persistent objects.  
  
Session session = sessionFactory.openSession();

**Session interface role**:

* Wraps a JDBC connection
* Factory for Transaction
* Holds a mandatory (first-level) cache of persistent objects, used when navigating the object graph or looking up objects by identifier

**12.What role does the SessionFactory interface play in Hibernate?**

The application obtains Session instances from a SessionFactory. There is typically a single SessionFactory for the whole applicationå¹¼reated during application initialization. The SessionFactory caches generate SQL statements and other mapping metadata that Hibernate uses at runtime. It also holds cached data that has been read in one unit of work and may be reused in a future unit of work  
  
SessionFactory sessionFactory = configuration.buildSessionFactory();

**13.What is the general flow of Hibernate communication with RDBMS?**

The general flow of Hibernate communication with RDBMS is :

* Load the Hibernate configuration file and create configuration object. It will automatically load all hbm mapping files
* Create session factory from configuration object
* Get one session from this session factory
* Create HQL Query
* Execute query to get list containing Java objects

**14.What is Hibernate Query Language (HQL)?**

Hibernate offers a query language that embodies a very powerful and flexible mechanism to query, store, update, and retrieve objects from a database. This language, the Hibernate query Language (HQL), is an object-oriented extension to SQL.

**15.How do you map Java Objects with Database tables?**

* First we need to write Java domain objects (beans with setter and getter).
* Write hbm.xml, where we map java class to table and database columns to Java class variables.

**Example** :

<hibernate-mapping>  
  <class name="com.test.User"  table="user">  
   <property  column="USER\_NAME" length="255"   
      name="userName" not-null="true"  type="java.lang.String"/>  
   <property  column="USER\_PASSWORD" length="255"  
 name="userPassword" not-null="true"  type="java.lang.String"/>  
 </class>  
</hibernate-mapping>

**16.What’s the difference between load() and get()?**

|  |  |
| --- | --- |
| **load()** | **get()** |
| Only use the load() method if you are sure that the object exists. | If you are not sure that the object exists, then use one of the get() methods. |
| load() method will throw an exception if the unique id is not found in the database. | get() method will return null if the unique id is not found in the database. |
| load() just returns a proxy by default and database won’t be hit until the proxy is first invoked. | get() will hit the database immediately. |

**17.What is the difference between and merge and update ?**

Use update() if you are sure that the session does not contain an already persistent instance with the same identifier, and merge() if you want to merge your modifications at any time without consideration of the state of the session.

**18.How do you define sequence generated primary key in hibernate?**

**19.Define cascade and inverse option in one-many mapping?**

cascade - enable operations to cascade to child entities.  
cascade="all|none|save-update|delete|all-delete-orphan"  
  
inverse - mark this collection as the "inverse" end of a bidirectional association.  
inverse="true|false"   
Essentially "inverse" indicates which end of a relationship should be ignored, so when persisting a parent who has a collection of children, should you ask the parent for its list of children, or ask the children who the parents are?

**22.Explain Criteria API**

Criteria is a simplified API for retrieving entities by composing Criterion objects. This is a very convenient approach for functionality like "search" screens where there is a variable number of conditions to be placed upon the result set.  
**Example** :

List employees = session.createCriteria(Employee.class)  
        .add(Restrictions.like("name", "a%") )  
          .add(Restrictions.like("address", "Boston"))  
 .addOrder(Order.asc("name") )  
 .list();

**23.Define HibernateTemplate?**

org.springframework.orm.hibernate.HibernateTemplate is a helper class which provides different methods for querying/retrieving data from the database. It also converts checked HibernateExceptions into unchecked DataAccessExceptions.

**24.What are the benefits does HibernateTemplate provide?**

The benefits of HibernateTemplate are :

* HibernateTemplate, a Spring Template class simplifies interactions with Hibernate Session.
* Common functions are simplified to single method calls.
* Sessions are automatically closed.
* Exceptions are automatically caught and converted to runtime exceptions.

**25.How do you switch between relational databases without code changes?**

Using Hibernate SQL Dialects , we can switch databases. Hibernate will generate appropriate hql queries based on the dialect defined.

**26.If you want to see the Hibernate generated SQL statements on console, what should we do?**

In Hibernate configuration file set as follows:   
<property name="show\_sql">true</property>

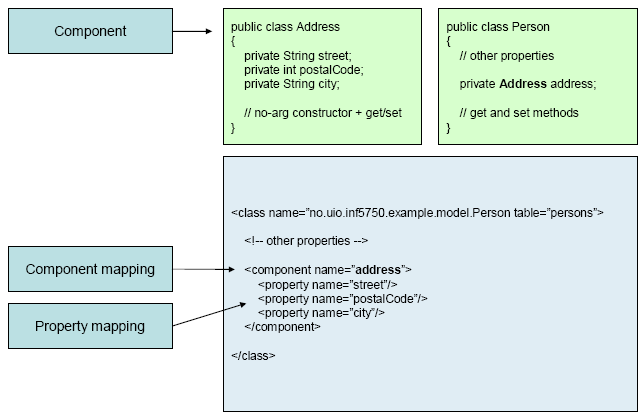
**27.What are derived properties?**

The properties that are not mapped to a column, but calculated at runtime by evaluation of an expression are called derived properties. The expression can be defined using the formula attribute of the element.

**28.What is component mapping in Hibernate?**

* A component is an object saved as a value, not as a reference
* A component can be saved directly without needing to declare interfaces or identifier properties
* Required to define an empty constructor
* Shared references not supported

**Example**:

  
 **29.What is the difference between sorted and ordered collection in hibernate?**

**sorted collection vs. order collection** :-

|  |  |
| --- | --- |
| **sorted collection** | **order collection** |
| A sorted collection is sorting a collection by utilizing the sorting features provided by the Java collections framework. The sorting occurs in the memory of JVM which running Hibernate, after the data being read from database using java comparator. | Order collection is sorting a collection by specifying the order-by clause for sorting this collection when retrieval. |
| If your collection is not large, it will be more efficient way to sort it. | If your collection is very large, it will be more efficient way to sort it . |

**32.What are the Collection types in Hibernate ?**

* Bag
* Set
* List
* Array
* Map

**33.What are the ways to express joins in HQL?**

HQL provides four ways of expressing (inner and outer) joins:-

* An *implicit* association join
* An ordinary join in the FROM clause
* A fetch join in the FROM clause.
* A *theta-style* join in the WHERE clause.

**34.Define cascade and inverse option in one-many mapping?**

cascade - enable operations to cascade to child entities.  
cascade="all|none|save-update|delete|all-delete-orphan"  
  
inverse - mark this collection as the "inverse" end of a bidirectional association.  
inverse="true|false"   
Essentially "inverse" indicates which end of a relationship should be ignored, so when persisting a parent who has a collection of children, should you ask the parent for its list of children, or ask the children who the parents are?

**35.What is Hibernate proxy?**

The proxy attribute enables lazy initialization of persistent instances of the class. Hibernate will initially return CGLIB proxies which implement the named interface. The actual persistent object will be loaded when a method of the proxy is invoked.

**36.How can Hibernate be configured to access an instance variable directly and not through a setter method ?**

By mapping the property with access="field" in Hibernate metadata. This forces hibernate to bypass the setter method and access the instance variable directly while initializing a newly loaded object.

**37.How can a whole class be mapped as immutable?**

@Entity

**@Immutable**

@Table(name = "stock", catalog = "mkyong")

**public** **class** Stock **implements** java.io.Serializable {

...

@OneToMany(fetch = FetchType.LAZY, mappedBy = "stock")

**@Immutable**

**public** Set<StockDailyRecord> getStockDailyRecords() {

**return** **this**.stockDailyRecords;

}

**38.What is the use of dynamic-insert and dynamic-update attributes in a class mapping?**

Criteria is a simplified API for retrieving entities by composing Criterion objects. This is a very convenient approach for functionality like "search" screens where there is a variable number of conditions to be placed upon the result set.

* dynamic-update (defaults to false): Specifies that UPDATE SQL should be generated at runtime and contain only those columns whose values have changed
* dynamic-insert (defaults to false): Specifies that INSERT SQL should be generated at runtime and contain only the columns whose values are not null.

**39.What do you mean by fetching strategy ?**

A *fetching strategy* is the strategy Hibernate will use for retrieving associated objects if the application needs to navigate the association.

**40.What is automatic dirty checking?**

Automatic dirty checking is a feature that saves us the effort of explicitly asking Hibernate to update the database when we modify the state of an object inside a transaction.

**41.What is transactional write-behind?**

Hibernate uses a sophisticated algorithm to determine an efficient ordering that avoids database foreign key constraint violations but is still sufficiently predictable to the user. This feature is called transactional write-behind.

|  |
| --- |
|  |

**42.What are Callback interfaces?**

Callback interfaces allow the application to receive a notification when something interesting happens to an object—for example, when an object is loaded, saved, or deleted. Hibernate applications don't need to implement these callbacks, but they're useful for implementing certain kinds of generic functionality.

**45.What are the types of inheritance models in Hibernate?**

There are three types of inheritance models in Hibernate:

* SINGLE TABLE
* TABLE PER CLASS
* JOINED