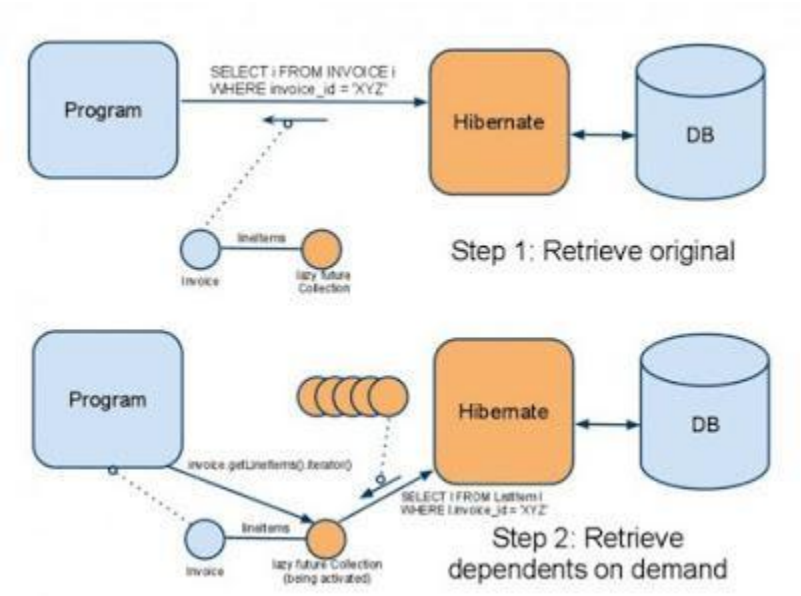
1. What is the difference between get and load in Hibernate?

get vs load is one of the most frequently asked Hibernate Interview questions, since the correct understanding of both `get()` and `load()` is required to effectively using Hibernate. Main difference between get and load is that, 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 the object is not found in the cache and only hit database if any method other than `getId()` is called. This can save a lot of performance in some cases. You can also see a difference between get and load in Hibernate for more differences and detailed discussion on this question.



2. What is the difference between save, persist and saveOrUpdate methods in Hibernate?

After get vs load, this is another Hibernate Interview question which appears quite often. All three methods i.e. `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.

Also, the 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 for complete differences between them in hibernate.

3. 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 a mapping document using `<sql-query>` tag and called using `Session.getNamedQuery()` method.

- Named query allows you to refer to 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 a single named query and `@NameQueries` is used to define multiple named query in hibernate. See Java Persistence with Hibernate for more details.

4. What is SessionFactory in Hibernate? Is SessionFactory thread-safe?

- Another common Interview question related to the Hibernate framework. `SessionFactory`, as the name suggests, is a factory to hibernate `Session` objects. SessionFactory is often built during start-up and used by application code to get the session object. It acts as a single data store and it's also thread-safe so that multiple threads can use the same `SessionFactory`.
- Usually, a Java JEE application has just one SessionFactory, and individual threads, which are servicing client's requests obtain hibernate Session instances from this factory, that's why any implementation of `SessionFactory` interface must be thread-safe.
- Also, the internal state of SessionFactory, which contains all metadata about Object/Relational mapping is Immutable and can not be changed once created.

5. What is Session in Hibernate? Can we share a single Session among multiple threads in Hibernate?

- This is usually asked as a follow-up question of the previous Hibernate Interview question. After `SessionFactory` its time for `Session`.
- `Session` represents a small unit of work in Hibernate, they maintain a connection with the database and they are **not thread-safe**, it means you can not share Hibernate Session between multiple threads. Though `Session` obtains database connection lazily it's good to close the session as soon as you are done with it.

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

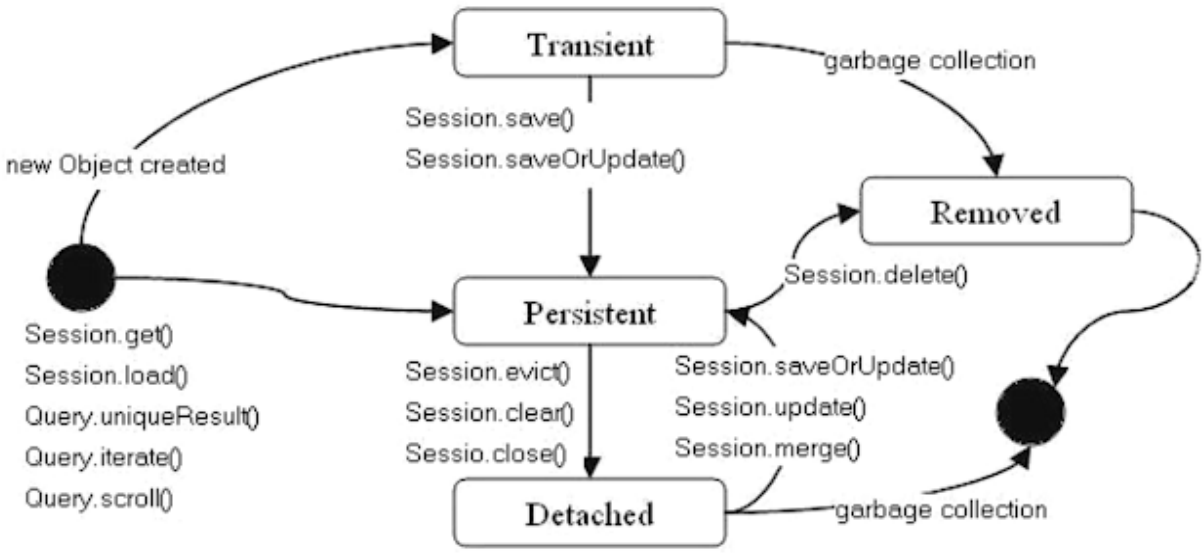
This is one of the easy Hibernate interview questions you ever face. A sorted collection is sorted in memory by using Java Comparator while an ordered collection uses the database's `order by` clause for ordering. For large data set it's better to use ordered collection to avoid any OutOfMemoryError in Java, by trying to sort them in memory.

7. What is the difference between a transient, persistent, and detached object in Hibernate?

In Hibernate, Objects can remain in three state `transient`, `persistent`, or `detached`. An object which is associated with the Hibernate session is called a `persistent` object.

Any change in this object will reflect in the database based on 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.



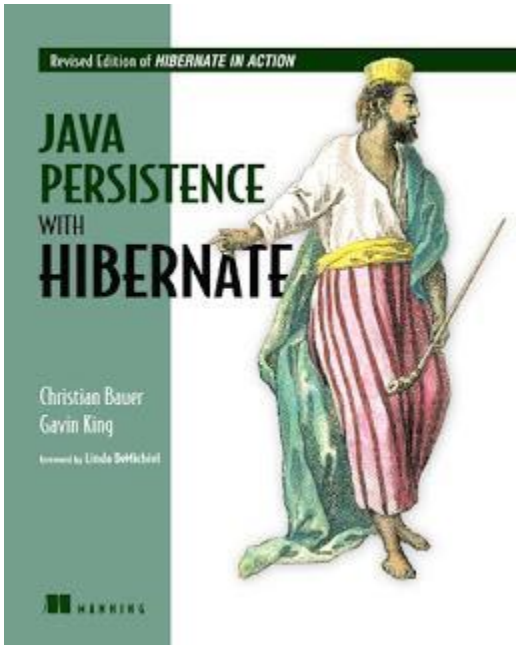
You can reattach detached objects to any other session by calling either `update()` or `saveOrUpdate()` method on that session. Transient objects have newly created an instance of persistence class, which is never associated with any Hibernate Session.

Similarly, you can call `persist()` or `save()` methods to make a transient object persistent. Just remember, here transient doesn't represent the transient keyword in Java, which is an altogether different thing.

8. What does Session lock() method do in Hibernate?

This one is one of the tricky Hibernate Interview questions because Session's `lock()` method reattach objects without synchronizing or updating with the database. So you need to be very careful while using the `lock()` method. By the way, you can always use the Session's `update()` method to sync with the database during attachment.

Sometimes this Hibernate question is also asked as *what is the difference between Session's lock() and update() method*. You can use this key point to answer that question as well. See Java Persistence with Hibernate for more details.



9. What is Second-level Cache in Hibernate?

This is one of the first interview question related to caching in Hibernate, you can expect a few more. Second-level Cache is maintained at `SessionFactory` level and can improve performance by saving a few database round trips. Another worth noting point is that second-level cache is available to the whole application rather than any particular session.

10. What is the query cache in Hibernate?

This question Sometimes asked as a follow-up of last Hibernate Interview question, `QueryCache` actually stores the 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.

11. Why it's important to provide no-argument constructor in Hibernate Entities?

Every Hibernate Entity class must contain a no-argument constructor, because Hibernate framework creates an instance of them using Reflection API, by calling `Class.newInstance()` method. This method will throw `InstantiationException` if it doesn't found any argument constructor inside Entity class.

12. Can we make a Hibernate Entity Class final?

Yes, you can make a Hibernate Entity class final, but that's not a good practice. Since Hibernate uses a proxy pattern for performance improvement in the case of the lazy association, by making an entity final, Hibernate will no longer be able to use a proxy, because Java doesn't allow extension of the final class, thus limiting your performance improvement options.

Though, you can avoid this penalty if your persistent class is an implementation of an interface, which declares all public methods defined in the Entity class.