**Hibernate**

**What is difference between Hibernate Session get() and load() method?**

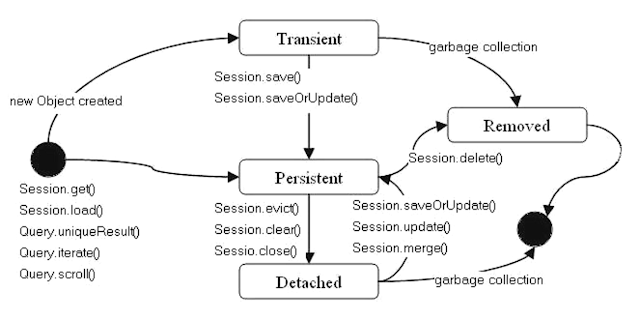
Hibernate session comes with different methods to load data from database

1. Session.load() – It will **always return a “proxy” (Hibernate  term) without hitting the database**. In Hibernate, proxy is an object  with the given identifier value, its properties are not initialized yet,  it just look like a temporary fake object. If no **row found , it will throws an ObjectNotFoundException**.
2. Session.get() – It **always hit the database and return the real object**, an object that represent the database row, not proxy.**If no row found , it return null**.
3. get() loads the data as soon as it’s called whereas load() returns a proxy object and loads data only when it’s actually required, **so load() is better because it support lazy loading.**
4. Since load() throws exception when data is not found, **we should use load only when we know data exists**.
5. We **should use get() when we want to make sure data exists in the database**.
6. Load can return proxy without hitting the database unless required (when you access any attribute other than id) but get() always go to the database, **so sometimes using load() can be faster than the get() method.**

[**https://www.journaldev.com/3472/hibernate-session-get-vs-load-difference-with-examples**](https://www.journaldev.com/3472/hibernate-session-get-vs-load-difference-with-examples)

**Difference between save() and saveOrUpdate()**

1. The main difference between save() and saveOrUpdate() method is that save() method performs an INSERT operation to store the object into the database, but INSERT will fail if the [primary key](http://java67.blogspot.com/2015/12/difference-between-primary-and-foreign.html) is already persistent i.e. object already exists in the database. This is why, you should only call save() with an absolutely new object which doesn't have any database identifier. Calling save() with the detached object will fail. This is opposite of saveOrUpdate() method, which can do either INSERT or UPDATE SQL query depending upon whether an object exists in the database or not. The saveOrUpdate() method first executes a [SELECT query](http://javarevisited.blogspot.com/2011/10/selct-command-sql-query-example.html) to determine if it needs to do an INSERT or UPDATE operation.
2. Another key difference between save() and saveOrUpdate() method is that former is used to bring a transient object to persistent state but saveOurUpdate() can bring both transient (new) and detached (existing) object into persistent state. It is often used to re-attach a detached object into Session



**Difference between save() and persist()**

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 differentiates 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.
4. Fourth difference between save and persist method in Hibernate is related to previous difference on save vs persist. Because of its above behaviour of persist method outside transaction boundary, its useful in long-running conversations with an extended Session context. On the other hand save method is not good in a long-running conversation with an extended Session context.
5. persist() – Hibernate persist is similar to save (with transaction) and it adds the entity object to the persistent context, so any further changes are tracked. If the object properties are changed before the transaction is committed or session is flushed, it will also be saved into database.
6. persist() also guarantees that it will not execute an INSERT statement if it is called outside of transaction boundaries. This is useful in long-running conversations with an extended Session/persistence context.
7. save() 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, not “sequence”), this INSERT happens immediately, no matter if you are inside or outside of a transaction. This is not good in a long-running conversation with an extended Session/persistence context.
8. persist() is supported by JPA, while save() is only supported by Hibernate.

**What’s the use of session.lock() in hibernate?**

session.lock() method of session class is used to reattach an object which has been detached earlier. This method of reattaching doesn’t check for any data synchronization in database while reattaching the object and hence may lead to lack of synchronization in data.

**Merge vs Update**

Hibernate handles persisting any changes to objects in the session when the session is flushed. update can fail if an instance of the object is already in the session. Merge should be used in that case. It merges the changes of the detached object with an object in the session, if it exists.

1. Update: Suppose we are dealing with any employee object in the same session then we should use update() or saveOrUpdate() method.
2. Update: if you are sure that the session does not contains an already persistent instance with the same identifier,then use update to save the data in hibernate
3. Merge: Suppose we are creating a session and load an employee object. Now object in session cache. If we close the session at this point and we edit state of object and tried to save using update() it will throw exception. To make object persistent we need to open another session. Now we load same object again in current session. So if we want to update present object with previous object changes we have to use merge() method. Merge method will merge changes of both states of object and will save in database.
4. Merge: if you want to save your modifications at any time with out knowing about the state of an session, then use merge() in hibernate.
5. Hibernate handles persisting any changes to objects in the session when the session is flushed. update can fail if an instance of the object is already in the session. Merge should be used in that case. It merges the changes of the detached object with an object in the session, if it exists.

**What are different states of an entity bean in Hibernate?**

The Entity bean has three states:  
Transient Persistent Detached

1. Transient: Whenever we create a new object of Entity bean then we can say that is in Transient state, At that time any modification in the object state does not effect on database. New objects created in Java program but not associated with any hibernate Session are said to be in the transient state.
2. Persistent: Whenever the Object of entity bean associated with session we can say that is in persistent state, if any change in the object state, then that modification effects in database.
3. Detached: Whenever the object is removed from session then it enters in to detached state. Any modification on detached state object, does not effect in database.
4. Transient: When an object is never persisted or associated with any session, it’s in transient state. Transient instances may be made persistent by calling save(), persist() or saveOrUpdate(). Persistent instances may be made transient by calling delete().
5. Persistent: When an object is associated with a unique session, it’s in persistent state. Any instance returned by a get() or load() method is persistent.
6. Detached: When an object is previously persistent but not associated with any session, it’s in detached state. Detached instances may be made persistent by calling update(), saveOrUpdate(), lock() or replicate(). The state of a transient or detached instance may also be made persistent as a new persistent instance by calling merge().

**What’s general hibernate flow using RDBMS?**

General hibernate flow involving RDBMS is as follows:  
a. Load configuration file and create object of configuration class.  
b. Using configuration object, create sessionFactory object.  
c. From sessionFactory, get one session.  
d. Create HQL query.  
e. Execute HQL query and get the results. Results will be in the form of a list.

**What’s the usage of Configuration Interface in hibernate?**

1. used to configure hibernate.
2. used to bootstrap hibernate.
3. used to build an immutable org.hibernate.SessionFactory
4. Mapping documents of hibernate are located using this interface.
5. org.hibernate.cfg.Configuration also allows you to specify configuration properties
6. To create a session factory in hibernate, an object of configuration is created first which refers to the path of configuration file and then for that configuration, session factory is created as given in the example below:

|  |  |
| --- | --- |
|  | Configuration config = new Configuration();  config.addResource(&amp;amp;quot;myinstance/configuration.hbm.xml&amp;amp;quot;);  config.setProperties( System.getProperties() );  SessionFactory sessions = config.buildSessionFactory(); |
|  |  |

**Hibernate configuration file**

The configuration file contains a database of specific configurations and is used to initialize SessionFactory. In the configuration XML file, the user provides database credentials or JNDI (Java Naming and Directory Interface) resource information. The other important part of the Hibernate configuration file is Dialect information, that allows Hibernate to know the database type, mapping file or class details.

Hibernate also requires a set of configuration settings related to database and other related parameters. All such information is usually supplied as a standard Java properties file called hibernate.properties, or as an XML file named hibernate.cfg.xml.

**Is Hibernate configuration file mandatory?**

Basically you are setting all the required properties via your properties object so there is no real need to tell Hibernate to look for a hibernate.cfg.xml file which is exactly what the configure() method does. No, it’s not mandatory to use hibernate.cfg.xml. Just use .configure().

**How can you configure Hibernate?**

* There are **two** main ways to do so – using **XML** and then the **annotation** provided by Java.
* In Hibernate versions earlier than Hibernate 4.0, the only way to configure the framework was by XML.
* The option to do so with Java-based annotations only became available in versions 4.0 and later.

**State the role of SessionFactory interface plays in Hibernate.**

* An application obtains Session instances from a SessionFactory which is typically single for the whole application created during its 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();

**What is Hibernate SessionFactory and how to configure it?**

* SessionFactory is the factory class used to get the Session objects. SessionFactory is responsible to read the hibernate configuration parameters and connect to the database and provide Session objects. Usually an application has a single SessionFactory instance and threads servicing client requests obtain Session instances from this factory.
* The internal state of a SessionFactory is immutable. Once it is created this internal state is set. This internal state includes all of the metadata about Object/Relational Mapping.
* SessionFactory also provide methods to get the Class metadata and Statistics instance to get the stats of query executions, second level cache details etc.

**What is Hibernate Session and how to get it?**

* Hibernate Session is the interface between java application layer and hibernate. This is the core interface used to perform database operations. Lifecycle of a session is bound by the beginning and end of a transaction.
* Session provide methods to perform create, read, update and delete operations for a persistent object. We can execute HQL queries, SQL native queries and create criteria using Session object.

**What is different between Session and SessionFactory in Hibernate?**

* The main difference between Session and SessionFactory is that the former is a single-threaded, short-lived object while later is Immutable and shared by all Session.
* It also lives until the Hibernate is running. Another difference between Session and SessionFactory is that former provides first level cache while SessionFactory provides the Second level cache.

**What is difference between openSession and getCurrentSession?**

* Hibernate SessionFactory getCurrentSession() method returns the session bound to the context. But for this to work, we need to configure it in hibernate configuration file. Since this session object belongs to the hibernate context, we don’t need to close it. Once the session factory is closed, this session object gets closed.

<property name="hibernate.current\_session\_context\_class">thread</property>

* Hibernate SessionFactory openSession() method always opens a new session. We should close this session object once we are done with all the database operations. We should open a new session for each request in multi-threaded environment.
* There is another method openStatelessSession() that returns stateless session

**Name some important interfaces of Hibernate framework?**

**SessionFactory (org.hibernate.SessionFactory)**: SessionFactory is an [immutable](https://www.journaldev.com/129/how-to-create-immutable-class-in-java) thread-safe cache of compiled mappings for a single database. We need to initialize SessionFactory once and then we can cache and reuse it. SessionFactory instance is used to get the Session objects for database operations.

**Session (org.hibernate.Session):** Session is a single-threaded, short-lived object representing a conversation between the application and the persistent store. It wraps JDBC java.sql.Connection and works as a factory for org.hibernate.Transaction. We should open session only when it’s required and close it as soon as we are done using it. Session object is the interface between java application code and hibernate framework and provide methods for CRUD operations.

**Transaction (org.hibernate.Transaction):** Transaction is a single-threaded, short-lived object used by the application to specify atomic units of work. It abstracts the application from the underlying JDBC or JTA transaction. A org.hibernate.Session might span multiple org.hibernate.Transaction in some cases.

**Name some important annotations used for Hibernate mapping?**

Hibernate supports JPA annotations and it has some other annotations in org.hibernate.annotations package. Some of the important JPA and hibernate annotations used are:

1. javax.persistence.Entity: Used with model classes to specify that they are entity beans.
2. javax.persistence.Table: Used with entity beans to define the corresponding table name in database.
3. javax.persistence.Access: Used to define the access type, either field or property. Default value is field and if you want hibernate to use getter/setter methods then you need to set it to property.
4. javax.persistence.Id: Used to define the primary key in the entity bean.
5. javax.persistence.EmbeddedId: Used to define composite primary key in the entity bean.
6. javax.persistence.Column: Used to define the column name in database table.
7. javax.persistence.GeneratedValue: Used to define the strategy to be used for generation of primary key. Used in conjunction with javax.persistence.GenerationType enum.
8. javax.persistence.OneToOne: Used to define the one-to-one mapping between two entity beans. We have other similar annotations as OneToMany, ManyToOne and ManyToMany
9. org.hibernate.annotations.Cascade: Used to define the cascading between two entity beans, used with mappings. It works in conjunction with org.hibernate.annotations.CascadeType
10. javax.persistence.PrimaryKeyJoinColumn: Used to define the property for foreign key. Used with org.hibernate.annotations.GenericGenerator and org.hibernate.annotations.Parameter

**Which design patterns are used in Hibernate framework?**

Some of the [design patterns](https://www.journaldev.com/1827/java-design-patterns-example-tutorial) used in Hibernate Framework are:

* Domain Model Pattern – An object model of the domain that incorporates both behaviour and data.
* Data Mapper – A layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself.
* [Proxy Pattern](https://www.journaldev.com/1572/proxy-design-pattern) for lazy loading
* [Factory pattern](https://www.journaldev.com/1392/factory-design-pattern-in-java) in SessionFactory

**What is HibernateTemplate class?**

When Spring and Hibernate integration started, Spring ORM provided two helper classes – HibernateDaoSupport and HibernateTemplate. The reason to use them was to get the Session from Hibernate and get the benefit of Spring transaction management. However from Hibernate 3.0.1, we can use SessionFactory *getCurrentSession()* method to get the current session and use it to get the spring transaction management benefits. If you go through above examples, you will see how easy it is and that’s why we should not use these classes anymore.

One other benefit of HibernateTemplate was exception translation but that can be achieved easily by using @Repository annotation with service classes.

**Fetching Strategies :**

Fetch strategies can be declared in the O/R mapping metadata, or over-ridden by a particular HQL or Criteria query. Hibernate defines the following fetching strategies:

1. Join fetching
2. Select fetching
3. Subselect fetching
4. Batch fetching
5. Immediate fetching
6. Lazy collection fetching
7. “Extra-lazy” collection fetching
8. Proxy fetching
9. “No-proxy” fetching
10. Lazy attribute fetching

What is the use of dialect in hibernate?

Answer # Dialect means “the variant of a language”. Hibernate, as we know, is database agnostic. It can work with different databases. However, databases have proprietary extensions/native SQL variations, and set/sub-set of SQL standard implementations. Therefore at some point hibernate has to use database specific SQL.

 What is hibernate proxy and how it helps in lazy loading?

Answer # A proxy is a subclass implemented at runtime. Hibernate creates a proxy (a subclass of the class being fetched) instead of querying the database directly, and this proxy will load the “real” object from the database whenever one of its methods is called.

What is criteria in hibernate?

Answer # In Hibernate, the Criteria API helps us build criteria query objects dynamically. Criteria is a another technique of data retrieval apart from HQL and native SQL queries. The primary advantage of the Criteria API is that it is intuitively designed to manipulate data without using any hard-coded SQL statements.

Question # 22 What is the difference between lazy and eager loading in hibernate?

Answer # All data is fetched when eager marked data in the object when session is connected. However, in case of lazy loading strategy, lazy loading marked object does not retrieve data if session is disconnected (after session.close() statement). All that can be made by hibernate proxy.

What does hibernate.hbm2ddl.auto create means?

Answer # hibernate.hbm2ddl.auto. Automatically validates or exports schema DDL to the database when the SessionFactory is created. With create-drop , the database schema will be dropped when the SessionFactory is closed explicitly.

How does Hibernate proxy work?

Answer # When a method is invoked on the object, Hibernate will fetch the data from the column and populate the object. This is the proxy mechanism. To add this new behavior (the loading of the data when a method is invoked), Hibernate will create a dynamic subclass of Person using CGLib and add the desired functionality.

Question # 29 What is first level cache in hibernate?

Answer # First level cache is associated with “session” object. The scope of cache objects is of session. First level cache is enabled by default and you can not disable it. When we query an entity first time, it is retrieved from database and stored in first level cache associated with hibernate session.

What is lazy fetching in Hibernate?

Answer # Lazy fetching decides whether to load child objects while loading the Parent Object. You need to do this setting respective hibernate mapping file of the parent class. Lazy = true (means not to load child) By default the lazy loading of the child objects is true.

Question # 32 What is an SQL dialect?

Answer # SQL Dialect. The SQL dialect, derived from the Structured Query Language, uses human-readable expressions to define query statements. Use a SQL query statement with the following ADSI search interfaces: The ActiveX Data Object (ADO) interfaces, which are Automation interfaces that use OLE DB.

Question # 33 What is in HQL?

Answer # Hibernate Query Language (HQL) is an object-oriented query language, similar to SQL, but instead of operating on tables and columns, HQL works with persistent objects and their properties. HQL queries are translated by Hibernate into conventional SQL queries, which in turns perform action on database.

Question # 34 What is the use of projection in hibernate?

Answer # To put it simple, Hibernate Projections are used in order to query only a subset of the attributes of an entity or group of entities you’re querying with Criteria. You can also use Projections to specify distinct clauses and aggregate functions like max , sum and so on. Like modifying the select clause in an SQL query.

Question # 35 What is Lazyinitializationexception in hibernate?

Answer # Indicates access to unfetched data outside of a session context. For example, when an uninitialized proxy or collection is accessed after the session was closed.

What is hbm2ddl in hibernate?

Answer # hibernate.hbm2ddl.auto Automatically validates or exports schema DDL to the database when the SessionFactory is created. With create-drop, the database schema will be dropped when the SessionFactory is closed explicitly.

What is the use of bag in hibernate?

Answer # Hibernate Bag is a java collection that stores elements without caring about the sequencing, but allow duplicate elements in the list. A bag is a random grouping of the objects in the list.

Question # 44 What is the use of Mappedby in hibernate?

Answer # With the mappedBy , you directly tell Hibernate/JPA that one table owns the relationship, and therefore it is stored as a column of that table. Without, the relationship is external and Hibernate/JPA need to create another table to store the relationship.

Question # 45 What is inverse true in hibernate?

Answer # The real meaning is that it defines which side is the parent or the relationship owner for the two entities (parent or child). Hence, inverse=”true” in a [Hibernate mapping](https://github.com/hibernate/hibernate-orm) shows that this class (the one with this XML definition) is the relationship owner; while the other class is the child.

Question # 46 What is a bidirectional relationship?

Answer # Bidirectional Relationships. In a bidirectional relationship, each entity has a relationship field or property that refers to the other entity. Through the relationship field or property, an entity class’s code can access its related object.

Question # 47 What is the dirty checking in hibernate?

Answer # Hibernate allows dirty checking feature.All persistent objects are monitored by hibernate.it detects which objects have been modified and then calls update statements on all updated objects.the process of updating the changed object is called automatic dirty checking.

What are the important interfaces in hibernate?

Answer #  
Session InterfaceSession Factory InterfaceConfiguration InterfaceTransaction InterfaceQuery and Criteria Interface

 What is difference between openSession() and getCurrentSession()?

Answer # If you set hibernate.current\_session\_context\_class to thread and then implement something like a servlet filter that opens the session – then you can access that session anywhere else by using the SessionFactory.getCurrentSession().

SessionFactory.openSession() always opens a new session that you have to close once you are done with the operations.

SessionFactory.getCurrentSession() returns a session bound to a context – you don’t need to close this.

What are the collection types in Hibernate?

Answer # Hibernate collections types are:  
java.util.List.java.util.Set.java.util.SortedSet.java.util.Map.java.util.SortedMap.java.util.Collection.or write the implementation of org.hibernate.usertype.UserCollectionType.

Question # 57 How to implement Joins in Hibernate?

Answer # Using HQL we can implement joins in hibernate.  
HQL Joins – HQL supports inner join, left outer join, right outer join and full join. For example, select e.name, a.city from Employee e INNER JOIN e.address a . In this query, Employee class should have a variable named address.

Question # 58 What is Named SQL Query?

Answer # A named query is a SQL expression represented as a table. In a named query, you can specify an SQL expression to select rows and columns returned from one or more tables in one or more data sources.

Question # 59 What is Hibernate Criteria API?

Answer # In Hibernate, the Criteria API helps us build criteria query objects dynamically. Criteria is a another technique of data retrieval apart from HQL and native SQL queries. The primary advantage of the Criteria API is that it is intuitively designed to manipulate data without using any hard-coded SQL statements.

Question # 60 What are callback interfaces in hibernate?

Answer # Callback interface for Hibernate code. To be used with HibernateTemplate ‘s execution methods, often as anonymous classes within a method implementation. The typical implementation will call Session.load/find/update to perform some operations on persistent objects.

What is transaction management in hibernate?

Answer # In a non-managed environment, Hibernate is usually responsible for its own database connection pool. The application developer has to manually set transaction boundaries (begin, commit, or rollback database transactions) themselves.

Question # 62 What are the mapping associations used in hibernate?

Answer # There are two mapping associations used in hibernate, they are:  
1) One-to-One Association2) Many-to-Many Association

Question # 63 What is Hibernate QBC API?

Answer # Hibernate Query By Criteria (QBC) API is used to create queries by manipulation of criteria objects at runtime.

Question # 64 What is hibernate criteria join?

Answer # Hibernate Criteria JOIN API allows users to perform join operation.  
Suppose you have to perform a operation like  
SELECT S.\*, C.\* FROM STUDENT S, CONTACT  C WHERE S.ROLL\_NO=C.ID;  
Then you can write this statement using Criteria join in a very simple way  
Criteria criteria = session.createCriteria(Student.class);criteria.setFetchMode(“Contact”, FetchMode.JOIN);List list = criteria.list();

Question # 65 What is the is the default transaction factory in hibernate?

Answer # JDBCTransactionFactory is the default transaction factory in hibernate.

66 What is JMX in Hybernate?

Answer # Java Management Extensions (JMX) is a Java technology that supplies tools for managing and monitoring applications, system objects, devices (such as printers) and service-oriented networks. Those resources are represented by objects called MBeans (for Managed Bean).

Question # 67 How to bind hibernate session factory to JNDI?

Answer # When binding the SessionFactory to JNDI, Hibernate will use the values of hibernate.jndi.url , hibernate.jndi.class to instantiate an initial context.

Question # 68 What are the fetching strategies in hibernate?

Answer # There are four fetching strategies

1. fetch-“join” = Disable the lazy loading, always load all the collections and entities.

2. fetch-“select” (default) = Lazy load all the collections and entities.

3. batch-size=”N” = Fetching up to ‘N’ collections or entities, \*Not record\*.

4. fetch-“subselect” = Group its collection into a sub select statement.

Question # 69 What are derived properties in hibernate?

Answer # In Hibernate a derived property (also called a calculated property) is a read-only property whose value is calculated at fetch time using SQL expressions.

Question # 70 What is version property in hibernate?

Answer # The <version> property (or @Version annotation) – We know that that Hibernate can provide optimistic locking through a version property on your persistent objects. Furthermore, the version property is automatically managed by Hibernate.

What does session lock () method do in hibernate?

Answer # The lock() method, with LockOptions.NONE, can be used to associate a detached object to a session and put the object back into a persistence context. On top is Hibernate code to reattach a detached object using a typical update method call. On the bottom is code to reattach a detached object using a lock method call.

Question # 72 What does evict do in hibernate?

Answer # evict() To detach the object from session cache, hibernate provides evict() method. After detaching the object from the session, any change to object will not be persisted. The associated objects will also be detached if the association is mapped with cascade=”evict”.

Question # 73 What is implicit polymorphism in hibernate?

Answer # Implicit polymorphism means if a class or interface is used in HQL, criteria or named queries, hibernate fetches the records from the table mapped to the used class along with all the tables mapped to its subclasses, at any hierarchy level. This is one of the great advantage for using hibernate.

Question # 74 What is table per concrete class in hibernate?

Answer # When we use Table Per Concrete class in hibernate, tables are created per class. So there are no nullable values in the table. Disadvantage of this approach is that duplicate columns are created in the subclass tables.

Question # 75 What is light object mapping in hibernate?

Answer # Light Object Mapping is one of the levels of ORM quality in which all entities are represented as classes and they are mapped manually in the Relational Tables.

**What is lazy loading?**  
Lazy loading is defined as a technique in which objects are loaded on an on-demand basis. It has been enabled by default since the advent of Hibernate 3 to ensure that child objects are not loaded when the parent is.  
  
**6. What are the ways to express joins in HQL?**  
  
HQL allows you to express joins in four ways:  
• An implicit association join  
• A fetch join in the FROM clause  
• A theta-style join in the WHERE clause  
• An ordinary join in the FROM clause  
  
**You may also like:**[Mistakes that could cost you your dream job](https://www.monsterindia.com/career-advice/3-mistakes-that-could-cost-you-your-dream-job-6590.html)  
  
**7. How would you define automatic dirty checking?**  
Automatic dirty checking can be defined as a feature that helps us in saving the effort of explicitly asking Hibernate to update the database every time we modify or make changes to the state of an object inside a transaction.  
  
**8. Explain the different ways Hibernate manages concurrency?**  
Hibernate has numerous ways of managing concurrency. They are as listed below:  
• Automatic versioning  
• Detached object  
• Extended user sessions  
  
**9. Can you detail out the various collection types in Hibernate?**  
There are five distinct collection types that are used in hibernate for one-to-many relationship mappings.  
• Bag  
• Set  
• List  
• Array  
• Map  
  
**10. Explain the difference between hibernate and Spring?**  
While Hibernate is an ORM framework tool used for data persistency, Spring, on the other hand, is a framework for enterprise applications. Moreover, Spring supports Hibernate and provides the different classes which are essentially templates containing the common code.  
  
In addition to the above-mentioned list of questions, here are a few more Frequently Asked Questions which we have put together to help you crack your all-important Hibernate interviews.  
  
• How many concurrent Hibernate sessions can exist at any point in time in an application?  
• Is SessionFactory a thread-safe object?  
• Can you explain the role Session interface plays in Hibernate?  
• Describe the process of switching between relational databases without code changes.  
• Explain the difference between the session.get() method and the session.load() method.  
• Is it possible to use Hibernate to map persistent entity POJO to XML files?  
• List out all the different states of an instance in Hibernate.  
• Can you describe the different contextual sessions in Hibernate?  
• Touch upon the pros of Hibernate Criteria API.  
• Explain the benefit of Hibernate Tools Eclipse plugin.  
• Describe component mapping in Hibernate.

## What is Casicadng?

<https://www.baeldung.com/jpa-cascade-types>

Entity relationships often depend on the existence of another entity — for example, the Person–Address relationship. Without the Person, the Address entity doesn't have any meaning of its own. When we delete the Person entity, our Address entity should also get deleted.

Cascading is the way to achieve this. **When we perform some action on the target entity, the same action will be applied to the associated entity.**

### 2.1. JPA Cascade Type

All JPA-specific cascade operations are represented by the javax.persistence.CascadeType enum containing entries:

* ALL
* PERSIST
* MERGE
* REMOVE
* REFRESH
* DETACH

### 2.2. Hibernate Cascade Type

Hibernate supports three additional Cascade Types along with those specified by JPA. These Hibernate-specific Cascade Types are available in org.hibernate.annotations.CascadeType:

* REPLICATE
* SAVE\_UPDATE
* LOCK

### What are best practices to follow with Hibernate framework?

Some of the best practices to follow in Hibernate are:

* + Always check the primary key field access, if it’s generated at the database layer then you should not have a setter for this.
  + By default hibernate set the field values directly, without using setters. So if you want hibernate to use setters, then make sure proper access is defined as @Access(value=AccessType.PROPERTY).
  + If access type is property, make sure annotations are used with getter methods and not setter methods. Avoid mixing of using annotations on both filed and getter methods.
  + Use native sql query only when it can’t be done using HQL, such as using database specific feature.
  + If you have to sort the collection, use ordered list rather than sorting it using Collection API.
  + Use named queries wisely, keep it at a single place for easy debugging. Use them for commonly used queries only. For entity specific query, you can keep them in the entity bean itself.
  + For web applications, always try to use JNDI DataSource rather than configuring to create connection in hibernate.
  + Avoid Many-to-Many relationships, it can be easily implemented using bidirectional One-to-Many and Many-to-One relationships.
  + For collections, try to use Lists, maps and sets. Avoid array because you don’t get benefit of lazy loading.
  + Do not treat exceptions as recoverable, roll back the Transaction and close the Session. If you do not do this, Hibernate cannot guarantee that in-memory state accurately represents the persistent state.
  + Prefer DAO pattern for exposing the different methods that can be used with entity bean
  + Prefer lazy fetching for associations