* **Lazy fetch vs eager fetch:**

[**https://howtoprogramwithjava.com/hibernate-eager-vs-lazy-fetch-type/**](https://howtoprogramwithjava.com/hibernate-eager-vs-lazy-fetch-type/)

[**http://stackoverflow.com/questions/17318340/how-hibernate-initialize-works**](http://stackoverflow.com/questions/17318340/how-hibernate-initialize-works)

* **Session vs session factory:**

[**http://way2java.com/hibernate/hibernate-first-level-and-second-level-cache-examples/**](http://way2java.com/hibernate/hibernate-first-level-and-second-level-cache-examples/)

Session factory creates session objects and manages them.

* **Hibernate Cascade Options – Cascade Attribute In Hibernate**
* **Difference between get and load in Hibernate.**

employee table has emp\_id, emp\_name, emp\_address, emp\_designation

sql query: select \* from employee where emp\_id=20

**using load...**will give a proxy object with

emp\_id=20,

emp\_name=null,

emp\_address=null,

emp\_designation=null.

The load() method may return a proxy instead of a real persistent instance.  
A proxy is a placeholder that triggers the loading of the real object when it’s   
accessed for the first time.

i.e sql query will be fired to hit DB only when this object is accessed for the first time. So basically load delays the hitting of DB until object is needed for the first time.

whereas **using get**... will give the below object.

emp\_id=20,

emp\_name=Neha,

emp\_address=valmikki st,

emp\_designation=PA.

in case of get, SQL query is fired immediately. It will hit DB each time.

Now, in case there is no record in DB for emp\_id 20,

get will return null

but load will return object not found exception. load throws an unrecoverable Exception when no Object is found for the ID.

Choosing between get() and load() is easy: If you’re certain the persistent   
object exists, and nonexistence would be considered exceptional, load() is a   
good option. If you aren’t certain there is a persistent instance with the given identifier, use get() and test the return value to see if it’s null. Using load() has a further implication: The application may retrieve a valid reference (a proxy) to a persistent instance without hitting the database to retrieve its persistent state. So load() might not throw an exception when it doesn’t find the persistent object in the cache or database; the exception would be thrown later, when the proxy is accessed.

https://ssl.gstatic.com/ui/v1/icons/mail/images/cleardot.gif

By far most important difference between get and load in my opinion. get method will return a completely initialized object if  Object is not on the cache but exists on [Database](http://javarevisited.blogspot.sg/2011/10/selct-command-sql-query-example.html), which may involve multiple round-trips to database based upon object relational mappings while load() method of Hibernate can return a **proxy** which can be initialized on demand (lazy initialization) when a non identifier method is accessed. Due to above reason use of load method will result in slightly **better performance**, but there is a caveat that proxy object will throw ObjectNotFoundException later if corresponding row doesn’t exists in database, instead of failing immediately so not a [fail fast](http://javarevisited.blogspot.sg/2012/02/fail-safe-vs-fail-fast-iterator-in-java.html) behavior.  
  
Read more: <http://javarevisited.blogspot.com/2012/07/hibernate-get-and-load-difference-interview-question.html#ixzz3QV5dfxz5>

* **Hibernate Mappings**

Parent – one, Child - many

**Hibernate one to one mapping:**  Each Student has a unique address.

Student Table: Student ID, Name, last name, phone, Address.

Address Table: Student ID, Address1, Address2, City, State

**Annotation:**

[**http://www.beingjavaguys.com/2013/09/hibernate-one-to-one-mapping.html**](http://www.beingjavaguys.com/2013/09/hibernate-one-to-one-mapping.html)

**XML configuration:**

[**http://examples.javacodegeeks.com/enterprise-java/hibernate/hibernate-one-to-one-relationship-example-xml-mapping-and-annotation/**](http://examples.javacodegeeks.com/enterprise-java/hibernate/hibernate-one-to-one-relationship-example-xml-mapping-and-annotation/)

**Hibernate one to many mapping: (always comes with many to one)**

1. Each Author has written multiple books

Author Table: Author ID, Author Name, Set<Books> (one to many) - one author to many books

Book Table: Book ID, Book Name, Author ID (many to one) – many books to one author

1. Each Vendor has multiple customers.

Vendor table: Vendor ID, Vendor Name, Set<Customers>(one to many)

Customer table: Customer ID, Customer Name, Vendor ID (many to one) - – give join column name

**Annotation:**

[**http://www.careerride.com/Hibernate-Interview-Questions.aspx**](http://www.careerride.com/Hibernate-Interview-Questions.aspx)

**XML configuration:**

[**http://www.java4s.com/hibernate/hibernate-one-to-many-bidirectional-mapping-example/**](http://www.java4s.com/hibernate/hibernate-one-to-many-bidirectional-mapping-example/)

**Hibernate many to one mapping:**

1. Each Vendor has multiple customers.

Vendor table: Vendor ID, Vendor Name

Customer table: Customer ID, Customer Name, Vendor (many to one)- – give join column name

1. Each User has multiple Vehicles

User Table: User ID, User Name

Vehicle Table: Vehicle ID, Vehicle Name, User (many to one) – give join column name– many vehicles to one user

**Annotation:**

[**http://www.dineshonjava.com/p/hibernate-many-to-one-mapping-tutorial.html#.VDdWeWeSwgs**](http://www.dineshonjava.com/p/hibernate-many-to-one-mapping-tutorial.html#.VDdWeWeSwgs)

**XML configuration:**

[**http://www.java4s.com/hibernate/hibernate-many-to-one-mapping-insert-query-example/**](http://www.java4s.com/hibernate/hibernate-many-to-one-mapping-insert-query-example/)

**Hibernate many to many mapping:**

Each Employee has multiple certificates. More than one employee can have the same certificate.

Employee table: Employee ID, Employee Name, Set<Certificates> - many to many – give join table name and – give join columns names

Certificate table: Certificate ID, Certificate Name

Joining table: Emp\_Cert Table: Employee ID, Certificate ID. Here primary key would be a composite key – (Employee ID, Certificate ID)

**Annotation:**

[**http://www.tutorialspoint.com/hibernate/hibernate\_many\_to\_many\_mapping.htm**](http://www.tutorialspoint.com/hibernate/hibernate_many_to_many_mapping.htm)

**XML configuration:**

[**http://www.beingjavaguys.com/2013/09/hibernate-many-to-many-example.html**](http://www.beingjavaguys.com/2013/09/hibernate-many-to-many-example.html)