1. **What’s Hibernate?**

Hibernate is an open-source and lightweight ORM tool that is used to store, manipulate and retrieve data from the database.

Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.

1. **What is ORM?**

ORM is an acronym for Object/Relational mapping. It is a programming strategy to map object with the data stored in the database. It simplifies data creation, data manipulation and data access.

1. **What are the core interfaces of Hibernate?**

* Configuration
* SessionFactory
* Session
* Query
* Criteria
* Transaction

1. **What is the difference between get() and load() method?**

|  |  |  |
| --- | --- | --- |
| **No.** | **get()** | **load()** |
| 1) | Returns **null** if object is not found. | Throws **ObjectNotFoundException** if object is not found. |
| 2) | get() method always **hit the database**. | load() method **doesn't hit** the database. |
| 3) | It returns real object **not proxy**. | It returns **proxy object.** |
| 4) | It should be used if **you are not sure** about the existence of instance. | It should be used if **you are sure** that instance exists. |

1. **What is the difference between session.save() and session.persist() method?**

|  |  |  |
| --- | --- | --- |
| **No.** | **save()** | **persist()** |
| 1) | returns the identifier (Serializable) of the instance. | return nothing because its return type is void. |
| 2) | Syn: public Serializable save(Object o) | Syn: public void persist(Object o) |

1. **What is the difference between update and merge method?**

|  |  |  |
| --- | --- | --- |
| **No.** | **update() method** | **merge() method** |
| 1) | Update means to edit something. | Merge means to combine something. |
| 2) | update() should be used if session doesn't contain an already persistent state with same id. It means update should be used inside the session only. After closing the session it will throw error. | merge() should be used if you don't know the state of the session, means you want to make modification at any time. |

Example: After closing session1, e1 is in detached state. It will not be in session1 cache. So if you call update() method, it will throw an error.

Then, we opened another session and loaded the same Employee instance. If we call merge in session2, changes of e1 will be merged in e2.

1. **What are the states of object/entity bean in hibernate?**
2. **Transient**: The object is in transient state if it is just created but has no primary key (identifier) and not associated with session.
3. **Persistent**: The object is in persistent state if session is open, and you just saved the instance in the database or retrieved the instance from the database.
4. **Detached**: The object is in detached state if session is closed. After detached state, object comes to persistent state if you call lock() or update() method.
5. **What are the inheritance mapping strategies?**
6. Table per hierarchy
7. Table per concrete class
8. Table per subclass
9. **What is SessionFactory?**

* SessionFactory provides the instance of Session. It is a factory of Session. It holds the data of second level cache that is not enabled by default.
* The internal state of SessionFactory, which contains all metadata about Object/Relational mapping is Immutable and cannot be changed once created.
* **SessionFactory** is a **thread-safe object**, many threads cannot access it simultaneously.

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

* It maintains a connection between hibernate application and database.
* It provides methods to store, update, delete or fetch data from the database such as persist(), update(), delete(), load(), get() etc. It is a factory of Query, Criteria and Transaction i.e. it provides factory methods to return these instances.
* **Session** is **not a thread-safe object**, many threads can access it simultaneously.

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

Session's **lock**() method reattach object without synchronizing or updating with the database. Always use Session's **update**() method to sync with the database during reattachment.

1. **What is difference between Hibernate save(), saveOrUpdate() and persist() methods?**

* Hibernate **save** can be used to save entity to database. Problem with save() is that it can be invoked without a transaction and if we have mapping entities, then only the primary object gets saved causing data inconsistencies. Also save returns the generated id immediately.
* Hibernate **persist** is similar to save with transaction. I feel it’s better than save because we can’t use it outside the boundary of transaction, so all the object mappings are preserved. Also persist doesn’t return the generated id immediately, so data persistence happens when needed.
* Hibernate **saveOrUpdate** results into insert or update queries based on the provided data. If the data is present in the database, update query is executed. We can use saveOrUpdate() without transaction also, but again you will face the issues with mapped objects not getting saved if session is not flushed.

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

Named queries are SQL queries which are defined in mapping document using **<sql-query>** tag and called using Session.getNamedQuery() method.

Hibernate Named Query syntax is checked when the hibernate session factory is created, thus making the application fail fast in case of any error in the named queries.

Hibernate Named Query is global, means once defined it can be used throughout the application. However one of the major disadvantage of Named query is that it’s hard to debug, because we need to find out the location where it’s defined.

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.

|  |
| --- |
| [xml]  <sql-query name = “studentdetails”>  <return alias=”std”/>  SELECT std.STUDENT\_ID AS {std.STUDENT\_ID},  std.STUDENT\_DISCIPLINE AS {std.discipline}, FROM Student std WHERE std.NAME LIKE :name  </sql-query>  [/xml]  **Then this query can be called as follows:**  [java]  List students = session.getNamedQuery(&amp;quot;studentdetails&amp;quot;)  .setString(&amp;quot;TomBrady&amp;quot;, name)  .setMaxResults(50)  .list();  [/java] |

**@NameQuery** is used to define single named query and **@NameQueries** is used to define multiple named query in hibernate.

1. **Name the two types of collections in hibernate?**

a. Sorted Collection

b. Order Collection

When we use Collection API sorting algorithms to sort a collection, it’s called sorted list. For small collections, it’s not much of an overhead but for larger collections it can lead to slow performance and OutOfMemory errors. Also the entity beans should implement **Comparable** or **Comparator** interface for it to work.

If we are using Hibernate framework to load collection data from database, we can use it’s Criteria API to use “order by” clause to get ordered list. Below code snippet shows you how to get it.

List<Employee> empList = session.createCriteria(Employee.class)

.addOrder(Order.desc("id")).list();

**Ordered list is better than sorted list** because the actual sorting is done at database level that is fast and doesn’t cause memory issues.

1. **What are the collection types in Hibernate?**

There are five collection types in hibernate used for one-to-many relationship mappings.

1. Bag 2. Set 3. List 4. Array 5.Map
2. **How can we reduce database write action times in Hibernate?**

Hibernate provides dirty checking feature which can be used to reduce database write times. Dirty checking feature of hibernate updates only those fields which require a change while keeps others unchanged.

1. **What is automatic dirty checking in hibernate?**

The automatic dirty checking feature of hibernate, calls update statement automatically on the objects that are modified in a transaction. Example given below:

|  |
| --- |
| 1. ... 2. SessionFactory factory = cfg.buildSessionFactory(); 3. Session session1 = factory.openSession(); 4. Transaction tx=session2.beginTransaction(); 6. Employee e1 = (Employee) session1.get(Employee.**class**, Integer.valueOf(101)); 8. e1.setSalary(70000); 10. tx.commit(); 11. session1.close(); |

Here, after getting employee instance e1 and we are changing the state of e1.

After changing the state, we are committing the transaction. In such case, state will be updated automatically. This is known as dirty checking in hibernate.

1. **What is lazy loading in hibernate?**

* Lazy loading in hibernate improves the performance. It loads the child objects on demand.
* Since Hibernate 3, lazy loading is enabled by default, you don't need to do lazy="true". It means not to load the child objects when parent is loaded.

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

Hibernate Query Language is known as an object oriented query language. It is like structured query language (SQL). It takes java objects in the same way as SQL takes tables. HQL is a Object Oriented Query language and is database independent.

1. **What is the difference between first level cache and second level cache?**

|  |  |  |
| --- | --- | --- |
| **No.** | **First Level Cache** | **Second Level Cache** |
| 1) | First Level Cache is **associated with Session**. | Second Level Cache is associated with **SessionFactory**. |
| 2) | It is **enabled** by default. | It is **not enabled** by default. |
| 3) | Mandatory cache through which all requests must pass | Optional cache and first-level cache will always be consulted before any attempt is made to locate an object in the second-level cache |
| 4) | Limited to a session | Available to whole application |

1. **What is Query level cache in hibernate?**

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.

This is an optional feature and requires two additional physical cache regions that hold the cached query results and the timestamps when a table was last updated. This is only useful for queries that are run frequently with the same parameters.

//Config File

<property name="hibernate.cache.use\_query\_cache">true</property>

//Code

Query query = session.createQuery("from Employee");

query.setCacheable(true);

query.setCacheRegion("ALL\_EMP");

1. **What is the benefit of Hibernate Criteria API?**

Hibernate Criteria API provides object oriented approach for querying the database and getting results. We can’t use Criteria in Hibernate to run update or delete queries or any DDL statements. Hibernate Criteria query is only used to fetch the results from the database using object oriented approach.

**Session.createCriteria** creates a new Criteria instance, for the given entity class, or a superclass of an entity class.

* Criteria API provides Projection that we can use for aggregate functions such as sum(), min(), max() etc.
* Criteria API can be used with ProjectionList to fetch selected columns only.
* Criteria API can be used for join queries by joining multiple tables, useful methods are createAlias(), setFetchMode() and setProjection()
* Criteria API provides addOrder() method that we can use for ordering the results.

1. **Why it's important to provide no argument constructor in Hibernate Entities?**

Hibernate uses **Reflection API** to create instance of Entity beans, usually when you call get() or load() methods. The method **Class.newInstance**() is used for this and it requires **no-args constructor**. So if you won’t have no-args constructor in entity beans, hibernate will fail to instantiate it and you will get **HibernateException**.

1. **Why we should not make Entity Class final?**

Hibernate use proxy classes for lazy loading of data, only when it’s needed. This is done by extending the entity bean, if the entity bean will be final then lazy loading will not be possible, hence low performance.

1. **What is the benefit of native sql query support in hibernate?**

Native SQL Query comes handy when we want to execute database specific queries that are not supported by Hibernate API such as query hints or the CONNECT keyword in Oracle Database.

1. **What is Hibernate Proxy and how it helps in lazy loading?**

* Hibernate uses proxy object to support lazy loading. Basically when you load data from tables, hibernate doesn’t load all the mapped objects.
* As soon as you reference a child or lookup object via getter methods, if the linked entity is not in the session cache, then the proxy code will go to the database and load the linked object.
* It uses javassist to effectively and dynamically generate sub-classed implementations of your entity objects.

1. **What are different ways to disable hibernate second level cache?**

* By setting use\_second\_level\_cache as false.
* By using CACHEMODE.IGNORE
* Using cache provider as org.hibernate.cache.NoCacheProvider

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

* **Domain Model Pattern** – An object model of the domain that incorporates both behavior 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** for lazy loading
* **Factory pattern** in SessionFactory

1. **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.
* One other benefit of HibernateTemplate was **exception translation** but that can be achieved easily by using **@Repository** annotation with service classes.

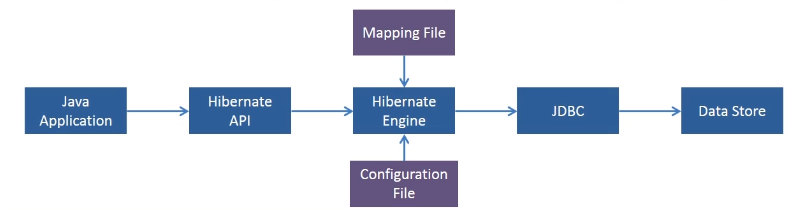
How to join two tables from a POJO in Hibernate?

|  |
| --- |
| @Table(name=”Person”)  **//specify secondary table annoatation along with FK**  @SecondaryTable(name=”person\_details”, pkJoinColumns(name=”person\_id”))  Public class Person |
| **//this is how we share data to secondary table**  @Column(name=”street\_address”, table=”person\_details”)  Private String streetAddress; |

How to create a composite key in Hibernate using multiple columns?

|  |
| --- |
| **First Approach**   1. Define a primary key class with embeddable notation which implements serializable interface 2. Then override equals & hashcode method 3. Then this class is used as an instance variable in our entity class which we annotate with **EmbeddedId** annotation.   ----------------Composite Key class------------------  @Embeddable  Public class ProvincePK implements Serializable{  Private String abbreviation;  Private String name;    //Getters & setters  }  ------------Entity class-------------------------  @Table(name=”province”)  Public class Province{    @EmbeddedId  Private ProvincePK pk = new ProvincePK();  //Getters & setters  } |
| **Second Approach**   1. Find a primary key class without annotations which implements serializable interface   -----------------------Primary Key Class---------------------------------  Public class **StatePK** implements Serializable{  Private String abbreviation;  Private String name;  //Getters & setters  }  --------------Entity Class----------------------  @Entity  @Table(name=”state”)  **@IdClass(StatePK.class)**  Public class **State**{  @Id  Private String abbreviation;  @Id  Private String name  } |

Hiberante Core API



HQL Criteria Query:

|  |
| --- |
| CriteriaBuilder builder = entityManager.getCriteriaBuilder();  CriteriaQuery<Person> criteria = builder.createQuery(Person.class);  Root<Person> person = criteria.form(Person.class);  Criteria.where(builder.equal(Person.get(Person\_provinceState),”OH”));  List<Person> persons = entityManager.createQuery(criteria).getResultList();  Syso(persons.size()); |