Top 50 Tricky Hibernate Interview Questions

FAQ's, Hibernate, Hibernate FAQ's

1. What is Hibernate?

Ans: Hibernate is a powerful, high performance object/relational persistence and query service. This lets the users to develop persistent classes following object-oriented principles such as association, inheritance, polymorphism, composition, and collections.

2. What is ORM ?

Ans: ORM stands for Object/Relational mapping. It is the programmed and translucent perseverance of objects in a Java application in to the tables of a relational database using the metadata that describes the mapping between the objects and the database. It works by transforming the data from one representation to another.

3. Why hibernate is advantageous over Entity Beans & JDBC?

Ans: An entity bean always works under the EJB container, which allows reusing of the object external to the container. An object can not be detached in entity beans and in hibernate detached objects are supported. Hibernate is not database dependent where as JDBC is database dependent. Query tuning is not needed for hibernate as JDBC is needed. Data can be placed in multiple cache which is supported by hibernate, whereas in JDBC the cache is to be implemented.

4. What are the different levels of ORM quality?

Ans: There are four levels defined for ORMquality.

- Pure relational
- Light object mapping
- Medium object mapping
- Full object mapping

5. How will you configure Hibernate?

Ans: The configuration files hibernate.cfg.xml (or hibernate.properties) and mapping files *.hbm.xml are used by the Configuration class to create (i.e. configure and bootstrap hibernate) the SessionFactory, which in turn creates the Session instances. Session instances are the primary interface for the persistence service.

" hibernate.cfg.xml (alternatively can use hibernate.properties): These two files are used to configure the hibernate sevice (connection driver class, connection URL, connection username, connection password, dialect etc). If both files are present in the classpath then hibernate.cfg.xml file overrides the settings found in the hibernate.properties file.

" Mapping files (*.hbm.xml): These files are used to map persistent objects to a relational database. It is the best practice to store each object in an individual mapping file (i.e mapping file per class) because storing large number of persistent classes into one mapping file can be difficult to manage and maintain. The naming convention is to use the same name as the persistent (POJO) class name. For example Account.class will have a mapping file named Account.hbm.xml. Alternatively hibernate annotations can be used as part of your persistent class code instead of the *.hbm.xml files.

6. What are derived properties?

Ans: 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.

7. Define HibernateTemplate?

Ans: *org.springframework.orm.hibernate.HibernateTemplate* is a helper class which provides different methods for querying/retrieving data from

Search:

Interview Question & Answers

- Tricky Core Java Interview Question and Answers
- Top 50 Tricky Hibernate Interview Question and Answers
- Servlet Interview Questions and Answers for Experienced Professionals
- Tricky SQL Queries

Popular Posts

- Top 50 Tricky Hibernate Interview Questions
- Tricky Core Java Interview Question with Answers
- Web Services Interview Questions and Answers
- Servlet Interview Questions and Answers for Experienced Professionals
- Accenture Java Interview Questions
- Most recently asked CapGemini Java Interview Questions and Answers

Recent Posts

Java Jobs

- jobs in Bangalore
- jobs in chennai
- jobs in cochin
- jobs in Delhi
- jobs in Gurgaon
- jobs in Hyderabad
- Jobs in Mumbai
- Jobs in Noida
- Jobs in Pune
- Onsite Jobs
- Other Locations
- Today Jobs

Walk-Ins

- Bangalore
- Chennai
- Hyderabad
- Kolkata
- Pune
- walkins

the database. It also converts checked HibernateExceptions into unchecked DataAccessExceptions.

8. What are the benefits does HibernateTemplate provide?
Ans: 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.

9. What is the difference between sorted and ordered collection in hibernate?
Ans: 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 .

10. What are the Collection types in Hibernate ?
Ans:

- Bag
- Set
- List
- Array
- Map

11. What is Hibernate proxy?
Ans: Mapping of classes can be made into a proxy instead of a table. A proxy is returned when actually a load is called on a session. The proxy contains actual method to load the data. The proxy is created by default by Hibernate, for mapping a class to a file. The code to invoke JDBC is contained in this class.

12. What is lazy fetching in hibernate ?
Ans: Lazy fetching is associated with child objects loading for its parents. While loading the parent, the selection of loading a child object is to be specified / mentioned in the hbm.xml file. Hibernate does not load the whole child objects by default. Lazy=true means not to load the child objects.

13. Explain the difference between transient and detached objects in hibernate ?
Ans: *Transient* objects do not have association with the databases and session objects. They are simple objects and not persisted to the database. Once the last reference is lost, that means the object itself is lost. And of course , garbage collected. The commits and rollbacks will have no effects on these objects. They can become into persistent objects through the save method calls of Session object.

The ***detached*** object have corresponding entries in the database. These are persistent and not connected to the Session object. These objects have the synchronized data with the database when the session was closed. Since then, the change may be done in the database which makes this object stale. The detached object can be reattached after certain time to another object in order to become persistent again.

14. What are the core interfaces of Hibernate framework ?
Ans:

- **Session Interface:** The basic interface for all hibernate applications. The instances are light weighted and can be created and destroyed without expensive process.
- **SessionFactory interface:** The delivery of session objects to hibernate applications is done by this interface. For the whole application, there will be generally one SessionFactory and can be shared by all the application threads.
- **Configuration Interface:** Hibernate bootstrap action is configured by this interface. The location specification is specified by specific mapping documents, is done by the instance of this interface.
- **Transaction Interface:** This is an optional interface. This interface is used to abstract the code from a transaction that is implemented such as a JDBC / JTA transaction.

- **Query and Criteria interface:** The queries from the user are allowed by this interface apart from controlling the flow of the query execution.

15. What are the different types of caches in Hibernate ?
Ans: Hibernate uses two different caches for objects: first-level cache and second-level cache. First-level cache is associated with the Session object, while second-level cache is associated with the SessionFactory object. By default, Hibernate uses first-level cache on a per-transaction basis. Hibernate uses this cache mainly to reduce the number of SQL queries it needs to generate within a given transaction.

16. How does the hibernate second-level cache work ?
Ans: Hibernate always tries to first retrieve objects from the session and if this fails it tries to retrieve them from the second-level cache. If this fails again, the objects are directly loaded from the database. Hibernate's static initialize() method, which populates a proxy object, will attempt to hit the second-level cache before going to the database. The Hibernate class provides static methods for manipulation of proxies.

17. What is a query cache in Hibernate ?
Ans: The query cache is responsible for caching the results and to be more precise the keys of the objects returned by queries. Let us have a look how Hibernate uses the query cache to retrieve objects. In order to make use of the query cache we have to modify the person loading example as follows.

```
Query query = session.createQuery("from Order as o where o.status=?");
query.setInt(0, "Active");
query.setCacheable(true); // the query is cacheable
List l = query.list();
```

18. What is callback interfaces in hibernate?
Ans: Callback interfaces of hibernate are useful in receiving event notifications from objects. For example, when an object is loaded or deleted, an event is generated and notification is sent using callback interfaces.

19. What is the difference between and merge and update ?
Ans: 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.

20. What is the difference between and load() and get() ?
Ans:
Load() method:

- Only use the load() method if you are sure that the object exists.
- load() method will throw an exception 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() method:

- If you are not sure that the object exists, then use one of the get()methods.
- get() will hit the database immediately.
- get() method will return null if the unique id is not found in the database.

21. What are the different caching strategies ?
Ans: Read-only: This strategy is useful for data that is read frequently but never updated. This is by far the simplest and best-performing cache strategy.
Read/write: Read/write caches may be appropriate if your data needs to be updated. They carry more overhead than read-only caches. In non-JTA environments, each transaction should be completed when Session.close() or Session.disconnect() is called.
Nonstrict read/write: This strategy does not guarantee that two transactions won't simultaneously modify the same data. Therefore, it may be most appropriate for data that is read often but only occasionally modified.
Transactional: This is a fully transactional cache that may be used only in a JTA environment.

22. What are the different fetching strategy in Hibernate ?
Ans: Hibernate3 defines the following fetching strategies:
Join fetching - Hibernate retrieves the associated instance or collection in the same SELECT, using an OUTER JOIN.
Select fetching - a second SELECT is used to retrieve the associated entity or collection. Unless you explicitly disable lazy fetching by specifying lazy="false", this second select will only be executed when you actually access the association.
Subselect fetching - a second SELECT is used to retrieve the associated collections for all entities retrieved in a previous query or fetch. Unless you explicitly disable lazy fetching by specifying lazy="false", this second select

will only be executed when you actually access the association.Batch fetching - an optimization strategy for select fetching - Hibernate retrieves a batch of entity instances or collections in a single SELECT, by specifying a list of primary keys or foreign keys.

23. What is version checking in Hibernate ?
Ans: version checking used in hibernate when more then one thread trying to access same data.
For example :
User A edit the row of the TABLE for update (In the User Interface changing data This is user thinking time) and in the same time User B edit the same record for update and click the update. Then User A click the Update and update done. Chnage made by user B is gone. In hibernate you can perevent slate object updatation using version checking.

Check the version of the row when you are upding the row. Get the version of the row when you are fetching the row of the TABLE for update. On the time of updation just fetch the version number and match with your version number (on the time of fetching).

24. Difference between getCurrentSession() and openSession() in Hibernate ?
Ans:
getCurrentSession():
The "current session" refers to a Hibernate Session bound by Hibernate behind the scenes, to the transaction scope. A Session is opened when getCurrentSession() is called for the first time and closed when the transaction ends.
openSession():
If you decide to use manage the Session yourself the go for sf.openSession(), you have to flush() and close() it. It does not flush and close() automatically.

25. What does session.refresh() do ?
Ans: It is possible to re-load an object and all its collections at any time, using the refresh() method. This is useful when database triggers are used to initialize some of the properties of the object.

26. How to get JDBC connections in hibernate ?
Ans: User Session.connection() method to get JDBC Connection.

27. What's the use of session.lock() in hibernate ?
Ans: 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.

28. What is the difference between the session.update() method and the session.lock() method ?
Ans: Both of these methods and saveOrUpdate() method are intended for reattaching a detached object. The session.lock() method simply reattaches the object to the session without checking or updating the database on the assumption that the database in sync with the detached object. It is the best practice to use either session.update(..) or session.saveOrUpdate(). Use session.lock() only if you are absolutely sure that the detached object is in sync with your detached object or if it does not matter because you will be overwriting all the columns that would have changed later on within the same transaction.

29. What is a SessionFactory?
Ans: The SessionFactory is the concept that is a single data store and thread safe. Because of this feature, many threads can access this concurrently and the sessions are requested, and also the cache that is immutable of compiled mappings for a specific database. A SessionFactory will be built only at the time of its startup. In order to access it in the application code, it should be wrapped in singleton. This wrapping makes the easy accessibility to it in an application code.

30. Explain about session interface?
Ans: This represents hibernate session which perform the manipulation on the database entities. Some of the activities performed by session interface are as follows they are managing the persistence state, fetching persisted ones and management of the transaction demarcation.

31. Explain about the dirty checking feature of Hibernate?
Ans: Dirty checking feature of the Hibernate allows users or developers to avoid time consuming data base write actions. This feature makes necessary updations and changes to the fields which require a change, remaining fields are left unchanged or untouched.

32. Explain the steps involved in creating database applications with Java using Hibernate?

Ans: Creating Database applications with Java is made simpler with Hibernate. First Plain old java object needs to be written, XML mapping file should be created which shows relationship between database and class attributes. Hibernate APIs can be used to store persistent objects.

33.Explain about hibernate.cfg.xml?
Ans: Hibernate can be configured with two types of files out of which hibernate.cfg.xmlis widely used and popular feature. Hibernate consults hibernate.cfg.xml file for its operating properties such as database dialect, connection string and mapping files. These files are searched on class path.

34. Explain about mapping description file?
Ans: Mapping description file is the second file which Hibernate uses to configure its functions. This mapping file has an extension *.hbm which instructs mapping between Java class and database tables. The usage of mapping description file rests entirely upon the business entity.

35. Explain about transaction file?
Ans: Transactions denote a work file which can save changes made or revert back the changes. A transaction can be started by session.beginTransaction() and it uses JDBC connection, CORBA or JTA. When this session starts several transactions may occur.

36. Explain about mapping files in Hibernate?
Ans: Mapping files forms the core of any database mapping tools. These files contain field to field mapping, usually this mapping occurs between classes and attributes. After mapping files they can be persist to the database. Tags can be used to indicate the presence of a primary key.

37. What is the effect when a transient mapped object is passed onto a Sessions save?
Ans: When a session.save( ) is passed to a transient mapped object it makes the method to become more persistent. Garbage collection and termination of the Java virtual machine stays as long as it is deleted explicitly. It may head back to its transient state.

38. Explain about version field?
Ans: Application level data integrity constants are important if you are making changes to offline information which is again backed by database. Higher level locking or versioning protocol is required to support them. Version field usage comes at this stage but the design and implementation process is left to the developer.

39. Explain State some advantages of hibernate?
Ans: Some of the advantages which a developer can get from Hibernate are as follows:
Mapping of one POJO table to one table is not required in hibernate.
It supports inheritance relationships and is generally a fast tool. Portability is necessary the greater benefit from hibernate. POJOs can be used in other applications where they are applicable.

40 Explain about addClass function?
Ans: This function translates a Java class name into file name. This translated file name is then loaded as an input stream from the Java class loader. This addclass function is important if you want efficient usage of classes in your code.

41. Explain about addjar() and addDirectory() methods?
Ans: These methods are the most convenient to use in hibernate. These methods allow you to load all your Hibernate documents at a time. These methods simplify code configuration, refactoring, layout, etc. These functions help you to add your hibernate mapping to Hibernate initialization files.

42. Explain about the id field?
Ans: This id field corresponds to the surrogate key which is generated by the database. These fields are handled by the id field. Name attribute is used to specify the names of the field and it should correspond to the method name of getid. This also should correspond to long type and the values should be stored I the database in the long column.

43. What are the most common methods of Hibernate configuration ?
Ans:The most common methods of Hibernate configuration are:
- Programmatic configuration
- XML configuration (hibernate.cfg.xml)

44. Define cascade and inverse option in one-many mapping ?
Ans:
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.

45. Explain Criteria API ?
Ans: 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();
```

46. How can a whole class be mapped as immutable ?
Ans: Mark the class as mutable="false" (Default is true),. This specifies that instances of the class are (not) mutable. Immutable classes, may not be updated or deleted by the application.

47. What is hibernate tuning ?
Ans: The key to obtain better performance in any hibernate application is to employ SQL Optimization, session management, and Data caching.

48. What is HQL ?
Ans: HQL stands for Hibernate Query Language. Hibernate allows the user to express queries in its own portable SQL extension and this is called as HQL. It also allows the user to express in native SQL.

49. How to Execute Stored procedure in Hibernate ?
Ans:

```
<sql-query name="selectAllEmployees_SP" callable="true">
<return alias="emp" class="employee">
<return-property name="empid" column="EMP_ID"/>
<return-property name="name" column="EMP_NAME"/>
<return-property name="address" column="EMP_ADDRESS"/>
{ ? = call selectAllEmployees() }
</return>
</sql-query>
```

Code:

```
SQLQuery sq = (SQLQuery)
session.getNamedQuery("selectAllEmployees_SP");
List results = sq.list();
```

50. What is a meant by light object mapping?
Ans: The entities are represented as classes that are mapped manually to the relational tables. The code is hidden from the business logic using specific design patterns. This approach is successful for applications with a less number of entities, or applications with common, metadata-driven data models.