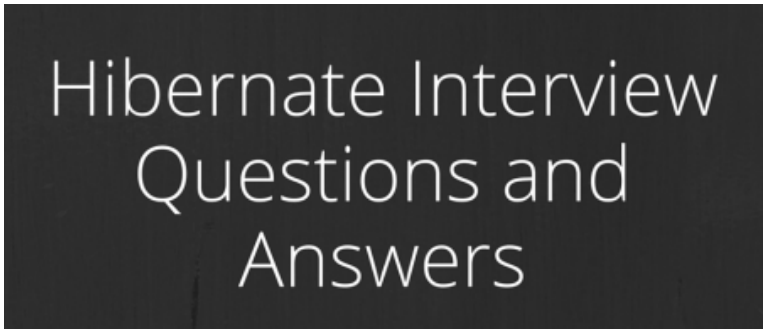
Hibernate Interview Questions and Answers

APRIL 2, 2018 BY PANKAJ — 81 COMMENTS

Hibernate is one of the most widely used ORM tool for Java applications. It's used a lot in enterprise applications for database operations. So I decided to write a post about **hibernate interview questions** to brush up your knowledge before the interview.

Whether you are fresher or experienced, having good knowledge or Hibernate ORM tool helps in cracking interview. Here I am providing important **hibernate interview questions** with answers to help you brush up your knowledge and impress your interviewer. Just like other interview questions posts, chances are that I will be adding more questions to this list in future, so you might want to bookmark it for future reference.



Recently I have written a lot of posts on hibernate, most of them contains complete downloadable projects. I will provide reference to them as and when needed and you can go through them to refresh your knowledge.

Hibernate Interview Questions

Hibernate Interview Questions and Answers

1. What is Hibernate Framework?

Object-relational mapping or ORM is the programming technique to map application domain model objects to the relational database tables. Hibernate is java based ORM tool that provides framework for mapping application domain objects to the relational database tables and vice versa.

Hibernate provides reference implementation of Java Persistence API, that makes it a great choice as ORM tool with benefits of loose coupling. We can use Hibernate persistence API for CRUD operations. Hibernate framework provide option to map plain old java objects to traditional database tables with the use of JPA annotations as well as XML based configuration.

Similarly hibernate configurations are flexible and can be done from XML configuration file as well as programmatically. For a quick overview of hibernate framework usage, you can go through Hibernate Beginners Tutorial.

2. What is Java Persistence API (JPA)?

Java Persistence API (JPA) provides specification for managing the relational data in applications. Current JPA version 2.1 was started in July 2011 as JSR 338. JPA 2.1 was approved as final on 22 May 2013.

JPA specifications is defined with annotations in javax.persistence package. Using JPA annotation helps us in writing implementation independent code.

3. What are the important benefits of using Hibernate Framework?

Some of the important benefits of using hibernate framework are:

1. Hibernate eliminates all the boiler-plate code that comes with JDBC and takes care of managing resources, so we can focus on business logic.
2. Hibernate framework provides support for XML as well as JPA annotations, that makes our code implementation independent.
3. Hibernate provides a powerful query language (HQL) that is similar to SQL. However, HQL is fully object-oriented and understands concepts like inheritance, polymorphism and association.
4. Hibernate is an open source project from Red Hat Community and used worldwide. This makes it a better choice than others because learning curve is small and there are tons of online documentations and help is easily available in forums.

5. Hibernate is easy to integrate with other Java EE frameworks, it's so popular that Spring Framework provides built-in support for integrating hibernate with Spring applications.
6. Hibernate supports lazy initialization using proxy objects and perform actual database queries only when it's required.
7. Hibernate cache helps us in getting better performance.
8. For database vendor specific feature, hibernate is suitable because we can also execute native sql queries.

Overall hibernate is the best choice in current market for ORM tool, it contains all the features that you will ever need in an ORM tool.

4. What are the advantages of Hibernate over JDBC?

Some of the important advantages of Hibernate framework over JDBC are:

1. Hibernate removes a lot of boiler-plate code that comes with JDBC API, the code looks more cleaner and readable.
2. Hibernate supports inheritance, associations and collections. These features are not present with JDBC API.
3. Hibernate implicitly provides transaction management, in fact most of the queries can't be executed outside transaction. In JDBC API, we need to write code for transaction management using commit and rollback. Read more at JDBC Transaction Management.
4. JDBC API throws `SQLException` that is a checked exception, so we need to write a lot of try-catch block code. Most of the times it's redundant in every JDBC call and used for transaction management. Hibernate wraps JDBC exceptions and throw `JDBCException` or `HibernateException` un-checked exception, so we don't need to write code to handle it. Hibernate built-in transaction management removes the usage of try-catch blocks.
5. Hibernate Query Language (HQL) is more object oriented and close to java programming language. For JDBC, we need to write native sql queries.
6. Hibernate supports caching that is better for performance, JDBC queries are not cached hence performance is low.
7. Hibernate provide option through which we can create database tables too, for JDBC tables must exist in the database.
8. Hibernate configuration helps us in using JDBC like connection as well as JNDI DataSource for connection pool. This is very important feature in enterprise application and completely missing in JDBC API.
9. Hibernate supports JPA annotations, so code is independent of implementation and easily replaceable with other ORM tools. JDBC code is very tightly coupled with the application.

5. Name some important interfaces of Hibernate framework?

Some of the important interfaces of Hibernate framework are:

1. **SessionFactory (org.hibernate.SessionFactory)**: SessionFactory is an immutable 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.

2. **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.
3. **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.

6. What is hibernate configuration file?

Hibernate configuration file contains database specific configurations and used to initialize SessionFactory. We provide database credentials or JNDI resource information in the hibernate configuration xml file. Some other important parts of hibernate configuration file is Dialect information, so that hibernate knows the database type and mapping file or class details.

7. What is hibernate mapping file?

Hibernate mapping file is used to define the entity bean fields and database table column mappings. We know that JPA annotations can be used for mapping but sometimes XML mapping file comes handy when we are using third party classes and we can't use annotations.

8. 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`

Here are two classes showing usage of these annotations.

```
package com.journaldev.hibernate.model;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;

@Entity
@Table(name = "EMPLOYEE")
@Access(value=AccessType.FIELD)
public class Employee {

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "emp_id")
        private long id;

        @Column(name = "emp_name")
```

```
package com.journaldev.hibernate.model;

import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.GenericGenerator;
import org.hibernate.annotations.Parameter;

@Entity
```

```
@Table(name = "ADDRESS")
@Access(value=AccessType.FIELD)
public class Address {

        @Id
        @Column(name = "emp_id", unique = true, nullable = false)
        @GeneratedValue(generator = "gen")
        @GenericGenerator(name = "gen", strategy = "foreign", parameters = {
@Parameter(name = "property", value = "employee") })
```

9. ## 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.

10. ## Hibernate SessionFactory is thread safe?

Internal state of SessionFactory is immutable, so it's thread safe. Multiple threads can access it simultaneously to get Session instances.

11. ## 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.

12. ## Hibernate Session is thread safe?

Hibernate Session object is not thread safe, every thread should get it's own session instance and close it after it's work is finished.

13. ## 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, for more details with examples please read Hibernate openSession vs getCurrentSession.

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

Hibernate session comes with different methods to load data from database. get and load are most used methods, at first look they seems similar but there are some differences between them.

1. 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.
2. Since load() throws exception when data is not found, we should use it only when we know data exists.
3. We should use get() when we want to make sure data exists in the database.

For clarification regarding the differences, please read Hibernate get vs load.

15. What is hibernate caching? Explain Hibernate first level cache?

As the name suggests, hibernate caches query data to make our application faster. Hibernate Cache can be very useful in gaining fast application performance if used correctly. The idea behind cache is to reduce the number of database queries, hence reducing the throughput time of the application.

Hibernate first level cache is associated with the Session object. Hibernate first level cache is enabled by default and there is no way to disable it. However hibernate provides methods through which we can delete selected objects from the cache or clear the cache completely.
Any object cached in a session will not be visible to other sessions and when the session is closed, all the cached objects will also be lost.

For better explanation, please read Hibernate First Level Cache.

16. How to configure Hibernate Second Level Cache using EHCache?

EHCache is the best choice for utilizing hibernate second level cache. Following steps are required to enable EHCache in hibernate application.

- Add hibernate-ehcache dependency in your maven project, if it's not maven then add corresponding jars.

```
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-ehcache</artifactId>
        <version>4.3.5.Final</version>
</dependency>
```

- Add below properties in hibernate configuration file.

```
<property
name="hibernate.cache.region.factory_class">org.hibernate.cache.ehcache.EhCacheF


<!-- For singleton factory -->
<!-- <property
name="hibernate.cache.region.factory_class">org.hibernate.cache.ehcache.Singletc

-->

<!-- enable second level cache and query cache -->
<property name="hibernate.cache.use_second_level_cache">true</property>
<property name="hibernate.cache.use_query_cache">true</property>
<property
name="net.sf.ehcache.configurationResourceName">/myehcache.xml</property>
```

- Create EHCache configuration file, a sample file myehcache.xml would look like below.

```
<?xml version="1.0" encoding="UTF-8"?>
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ehcache.xsd" updateCheck="true"
    monitoring="autodetect" dynamicConfig="true">

    <diskStore path="java.io.tmpdir/ehcache" />

    <defaultCache maxEntriesLocalHeap="10000" eternal="false"
        timeToIdleSeconds="120" timeToLiveSeconds="120"
diskSpoolBufferSizeMB="30"
        maxEntriesLocalDisk="10000000"
diskExpiryThreadIntervalSeconds="120"
        memoryStoreEvictionPolicy="LRU" statistics="true">
        <persistence strategy="localTempSwap" />
    </defaultCache>

    <cache name="employee" maxEntriesLocalHeap="10000" eternal="false"
        timeToIdleSeconds="5" timeToLiveSeconds="10">
```

```
        <persistence strategy="localTempSwap" />
    </cache>

    <cache name="org.hibernate.cache.internal.StandardQueryCache"
        maxEntriesLocalHeap="5" eternal="false" timeToLiveSeconds="120">
        <persistence strategy="localTempSwap" />
    </cache>
```

- Annotate entity beans with @Cache annotation and caching strategy to use. For example,

```
import org.hibernate.annotations.Cache;
import org.hibernate.annotations.CacheConcurrencyStrategy;

@Entity
@Table(name = "ADDRESS")
@Cache(usage=CacheConcurrencyStrategy.READ_ONLY, region="employee")
public class Address {

}
```

That's it, we are done. Hibernate will use the EHCache for second level caching, read Hibernate EHCache Example for a complete example with explanation.

17. What are different states of an entity bean?

An entity bean instance can exist is one of the three states.

1. **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().
2. **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.
3. **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().

18. What is use of Hibernate Session merge() call?

Hibernate merge can be used to update existing values, however this method create a copy from the passed entity object and return it. The returned object is part of persistent context and tracked for any changes, passed object is not tracked. For example program, read Hibernate merge.

19. 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. For example usage of these methods, read Hibernate save vs persist.

20. What will happen if we don't have no-args constructor in Entity bean?

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`.

21. What is difference between sorted collection and ordered collection, which one is better?

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, read more at java object list sorting.

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.

22. 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

23. How to implement Joins in Hibernate?

There are various ways to implement joins in hibernate.

- Using associations such as one-to-one, one-to-many etc.
- Using JOIN in the HQL query. There is another form "join fetch" to load associated data simultaneously, no lazy loading.
- We can fire native sql query and use join keyword.

24. 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.

25. What is HQL and what are it's benefits?

Hibernate Framework comes with a powerful object-oriented query language – Hibernate Query Language (HQL). It's very similar to SQL except that we use Objects instead of table names, that makes it more close to object oriented programming.

Hibernate query language is case-insensitive except for java class and variable names. So SeLeCT is the same as sELEct is the same as SELECT, but com.journaldev.model.Employee is not same as com.journaldev.model.EMPLOYEE.

The HQL queries are cached but we should avoid it as much as possible, otherwise we will have to take care of associations. However it's a better choice than native sql query because of Object-Oriented approach. Read more at HQL Example.

26. What is Query Cache in Hibernate?

Hibernate implements a cache region for queries resultset that integrates closely with the hibernate second-level cache.

This is an optional feature and requires additional steps in code. This is only useful for queries that are run frequently with the same parameters. First of all we need to configure below property in hibernate configuration file.

```
<property name="hibernate.cache.use_query_cache">true</property>
```

And in code, we need to use setCacheable(true) method of Query, quick example looks like below.

```
Query query = session.createQuery("from Employee");
query.setCacheable(true);
query.setCacheRegion("ALL_EMP");
```

27. Can we execute native sql query in hibernate?

Hibernate provide option to execute native SQL queries through the use of `SQLQuery` object.

For normal scenarios, it is however not the recommended approach because we loose benefits related to hibernate association and hibernate first level caching. Read more at Hibernate Native SQL Query Example.

28. 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.

29. What is Named SQL Query?

Hibernate provides Named Query that we can define at a central location and use them anywhere in the code. We can created named queries for both HQL and Native SQL.

Hibernate Named Queries can be defined in Hibernate mapping files or through the use of JPA annotations @NamedQuery and @NamedNativeQuery.

30. What are the benefits of Named SQL Query?

Hibernate Named Query helps us in grouping queries at a central location rather than letting them scattered all over the code.
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.

31. What is the benefit of Hibernate Criteria API?

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

Some of the common usage of Criteria API are:

- 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 can be used for fetching results with conditions, useful methods are add() where we can add Restrictions.
- Criteria API provides addOrder() method that we can use for ordering the results.

Learn some quick examples at Hibernate Criteria Example.

32. How to log hibernate generated sql queries in log files?

We can set below property for hibernate configuration to log SQL queries.

```
<property name="hibernate.show_sql">true</property>
```

However we should use it only in Development or Testing environment and turn it off in production environment.

33. 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.

34. How to implement relationships in hibernate?

We can easily implement one-to-one, one-to-many and many-to-many relationships in hibernate. It can be done using JPA annotations as well as XML based configurations. For better understanding, you should go through following tutorials.

1. Hibernate One to One Mapping
2. Hibernate One to Many Mapping
3. Hibernate Many to Many Mapping

35. How transaction management works in Hibernate?

Transaction management is very easy in hibernate because most of the operations are not permitted outside of a transaction. So after getting the session from SessionFactory, we can call session `beginTransaction()` to start the transaction. This method returns the Transaction reference that we can use later on to either commit or rollback the transaction.

Overall hibernate transaction management is better than JDBC transaction management because we don't need to rely on exceptions for rollback. Any exception thrown by session methods automatically rollback the transaction.

36. What is cascading and what are different types of cascading?

When we have relationship between entities, then we need to define how the different operations will affect the other entity. This is done by cascading and there are different types of it.

Here is a simple example of applying cascading between primary and secondary entities.

```
import org.hibernate.annotations.Cascade;

@Entity
@Table(name = "EMPLOYEE")
public class Employee {

@OneToOne(mappedBy = "employee")
@Cascade(value = org.hibernate.annotations.CascadeType.ALL)
private Address address;

}
```

Note that Hibernate CascadeType enum constants are little bit different from JPA `javax.persistence.CascadeType`, so we need to use the Hibernate CascadeType and Cascade annotations for mappings, as shown in above example.
Commonly used cascading types as defined in CascadeType enum are:

1. None: No Cascading, it's not a type but when we don't define any cascading then no operations in parent affects the child.
2. ALL: Cascades save, delete, update, evict, lock, replicate, merge, persist. Basically everything
3. SAVE_UPDATE: Cascades save and update, available only in hibernate.
4. DELETE: Corresponds to the Hibernate native DELETE action, only in hibernate.
5. DETATCH, MERGE, PERSIST, REFRESH and REMOVE – for similar operations
6. LOCK: Corresponds to the Hibernate native LOCK action.
7. REPLICATE: Corresponds to the Hibernate native REPLICATE action.

37. How to integrate log4j logging in hibernate application?

Hibernate 4 uses JBoss logging rather than slf4j used in earlier versions. For log4j configuration, we need to follow below steps.

- Add log4j dependencies for maven project, if not maven then add corresponding jar files.
- Create log4j.xml configuration file or log4j.properties file and keep it in the classpath. You can keep file name whatever you want because we will load it in next step.

- For standalone projects, use static block to configure log4j using `DOMConfigurator` or `PropertyConfigurator`. For web applications, you can use ServletContextListener to configure it.

That's it, our setup is ready. Create `org.apache.log4j.Logger` instance in the java classes and start logging. For complete example code, you should go through Hibernate log4j example and Servlet log4j example.

38. How to use application server JNDI DataSource with Hibernate framework?

For web applications, it's always best to allow servlet container to manage the connection pool. That's why we define JNDI resource for DataSource and we can use it in the web application. It's very easy to use in Hibernate, all we need is to remove all the database specific properties and use below property to provide the JNDI DataSource name.

```
<property
name="hibernate.connection.datasource">java:comp/env/jdbc/MyLocalDB</property>
```

For a complete example, go through Hibernate JNDI DataSource Example.

39. How to integrate Hibernate and Spring frameworks?

Spring is one of the most used Java EE Framework and Hibernate is the most popular ORM framework. That's why Spring Hibernate combination is used a lot in enterprise applications. The best part with using Spring is that it provides out-of-box integration support for Hibernate with **Spring ORM** module. Following steps are required to integrate Spring and Hibernate frameworks together.

1. Add hibernate-entitymanager, hibernate-core and spring-orm dependencies.
2. Create Model classes and corresponding DAO implementations for database operations. Note that DAO classes will use SessionFactory that will be injected by Spring Bean configuration.
3. If you are using Hibernate 3, you need to configure `org.springframework.orm.hibernate3.LocalSessionFactoryBean` or `org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean` in Spring Bean configuration file. For Hibernate 4, there is single class `org.springframework.orm.hibernate4.LocalSessionFactoryBean` that should be configured.
4. Note that we don't need to use Hibernate Transaction Management, we can leave it to Spring declarative transaction management using `@Transactional` annotation.

For complete example go through Spring Hibernate Integration and Spring MVC Hibernate Integration.

40. 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, shown in above spring mvc example. This is a trick question to judge your knowledge and whether you are aware of recent developments or not.

41. How to integrate Hibernate with Servlet or Struts2 web applications?

Hibernate integration with Servlet or Struts2 needs to be done using `ServletContextListener`, a complete example can be found at Hibernate Struts2 Integration Example.

42. Which design patterns are used in Hibernate framework?

Some of the design patterns used in Hibernate Framework are:

- 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

43. 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

44. What is Hibernate Validator Framework?

Data validation is integral part of any application. You will find data validation at presentation layer with the use of Javascript, then at the server side code before processing it. Also data validation occurs before persisting it, to make sure it follows the correct format.

Validation is a cross cutting task, so we should try to keep it apart from our business logic. That's why JSR303 and JSR349 provides specification for validating a bean by using annotations. Hibernate Validator provides the reference implementation of both these bean validation specs. Read more at Hibernate Validation Example.

45. What is the benefit of Hibernate Tools Eclipse plugin?

Hibernate Tools plugin helps us in writing hibernate configuration and mapping files easily. The major benefit is the content assist to help us with properties or xml tags to use. It also validates them against the Hibernate DTD files, so we know any mistakes before hand. Learn how to install and use at Hibernate Tools Eclipse Plugin.

That's all for **Hibernate Interview Questions and Answers**, I hope it will help you for interview as a fresher or experienced person. Please let me know if I have missed any important question here, I will add that to the list.

FILED UNDER: HIBERNATE, INTERVIEW QUESTIONS

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

« Hibernate Validator JSR303 Example Tutorial

Hibernate Tutorial »

Comments

Sanjay Panchal says

JUNE 25, 2018 AT 2:19 AM

Thanks a lot for your effort.

Reply

Abhishek says

JUNE 12, 2018 AT 3:49 AM

Good Explanation and lucrative and concise.

Reply

vk says
MAY 16, 2018 AT 7:23 AM
Very nice, great collection
Reply

test says
MARCH 13, 2018 AT 4:02 PM
include question about transaction isolation levels too.
Reply

Vikash says
JANUARY 27, 2018 AT 4:12 AM
Great Hibernate refresher, very crisp and concise, but still covers all major points.
Reply

Ramana Yalamanchili says
JANUARY 17, 2018 AT 11:09 AM
Pankaj,
Advantage 4 states – Hibernate wraps JDBC exceptions and throw JDBCException or HibernateException un-checked exception, so we don't need to write code to handle it.
I am surprised the HibernateException is unchecked. Oracle suggests otherwise. We should not consider this as an advantage of Hibernate. It is more like a disadvantage of Hibernate in the sense that any exceptions caused "Not because of bad programming" also go unchecked. If HibernateException is checked, client code which is calling the code which throws this exception can handle the exception better.
https://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html
Reply

sujit kumar nishank says
NOVEMBER 23, 2017 AT 6:48 AM
great job pankaj . nice blog and we can get some basic about HB

Reply

Rizwana says

NOVEMBER 15, 2017 AT 4:13 AM

Excellent work Pankaj and Kindly provide download option also, It will very helpful to us.

Reply

Ramesh Nayak says

OCTOBER 24, 2017 AT 10:16 AM

Thank You very much for such amazing tutorial and interview questions and along with answers with brief explanation.

Reply

Madhusudhan T says

SEPTEMBER 6, 2017 AT 11:10 AM

Excellent Tutorials.. Thanks Pankaj…

Reply

Sumana says

AUGUST 23, 2017 AT 5:27 AM

Hats off man.

All your tutorials are really awesome.

Thank you very much.

Reply

Archana says

AUGUST 19, 2017 AT 2:33 AM

Very useful information. This website just doesn't give direct answers instead provides enough explanation to understand the concepts. Really wonderful. Thanks for all your effort!

Reply

Sardar Saikh says
JULY 27, 2017 AT 8:14 AM
This blog is really helpful for interview preparation. Thanks a lot for your effort.
Reply

Anurag Singh says
JULY 26, 2017 AT 8:03 AM
this tutorial is very helpful.
thanks
Reply

harini says
JULY 5, 2017 AT 5:20 PM
Hi,
is there a questionarrie for jpa?
Reply

harini says
JULY 5, 2017 AT 5:19 PM
Hi,
is there questions for jpa which you had provided?
Reply

Malik says
MAY 18, 2017 AT 7:20 AM
Hi Pankaj , thanks for such great tutorial.
Wanted to check if its possible to implement LATERAL join in hibernate criteria.
let say for the following example, any idea how can be implement using hibernate criteria .
Hibernate version is below 5*
.
select tab1.Col1, tab2.Col2
from Table1 tab1
inner join Table2 tab2 on tab2.col2 = tab1.col1
inner join LATERAL
(

select tab3.col3 from Table3 tab3 where tab3.col3 = tab1.col1
)
union
(
select tab4.col4 from Table4 tab4 where tab4.col4 = tab1.col1
) tab5 on tab1.col1 = tab5.col1
inner join Table6 tab6 on tab1.col1 = tab6.col1
Reply

Hojat says
MAY 3, 2017 AT 1:05 AM
Good job man!
Reply

adiseshu says
APRIL 6, 2017 AT 11:51 PM
good to go through this before interviews
well done
thank u
adiseshu
Reply

Hitesh says
MARCH 10, 2017 AT 3:20 AM
Hello sir,
I am reading this question and answer it is very nice.,
Can you provide how to use hibernate in spring tool suite with example for better understand.
Reply

vaass says
JANUARY 4, 2017 AT 7:56 PM
best Q&A..Appreciate
Reply

Anil says
DECEMBER 26, 2016 AT 5:56 AM
As you said SessionFactory object is immutable i.e. thrad-safe that means multiple threads can access it simultaneously to get Session instances.
it's wrong, le me correct you, thread-safety means multiple thread can not access it simultaneously, at time only one thread can access it. If one thread is accessing the SessionFactory object that time another thread has to wait to access the same.
Reply

Makarand says
FEBRUARY 8, 2017 AT 1:00 AM
I think Pankaj is trying to say same thing but you put it in different way.which also correct.
Reply

Ahasan Siddiqui says
MARCH 3, 2017 AT 9:06 AM
Hi Makarand,
Actually If A Class object is immutable its mean several thread can access it simultaneously but they can perform the read operation at a time. no modification can be done by the thread. like by default String classes and Wrapper classes are immutable and so they are thread safe by default. because any thread are not able to modify the immutable object. only read access is possible. i.e. it is not doing any thread safety directly but it is thread safe.
Reply

Chirag says
DECEMBER 9, 2016 AT 10:09 AM
your question set for hibernate is really good. But would be great if you can include 1 more famous question to your list related to hibernate N+1 problem.
Thanks
Reply

Amit S says
DECEMBER 8, 2016 AT 10:19 AM
Nice article Pankaj. Great work!!!
Reply

Niraj IMS says
MAY 19, 2016 AT 3:13 AM
HibernateDaoSupport and HibernateTemplate should not be used for any further development.
This is what to be instructed in the latest document from Spring officials.
Do you have any idea as to why such prohibition?
Thanks for everything…
Reply

Priyesh says
FEBRUARY 12, 2016 AT 1:37 AM
Hi Pankaj,
Good article for interview preparation, but in question# 10 have little doubt if SessionFactory is thread safe than how Multiple threads can access it simultaneously to get Session instances.
Reply

Himansu Nayak says
AUGUST 21, 2016 AT 10:55 AM
SessionFactoryImpl class of hibernate is almost immutable but SessionBuilder is still mutable
Reply

Naveen says
SEPTEMBER 26, 2016 AT 4:28 AM
ThreadSafe means…the SessionFactory object's state cannot be changed by any thread accessing it.
It does not mean threads cant access simultaneously.
Multiple threads can safely access a ThreadSafe object-SessionFactory.
Reply

prakash says
OCTOBER 30, 2015 AT 2:53 AM
once the data is present(come to) in proxy object..then after the data was changed…then the same request is come…here question is the data is taken from proxy or database..
Reply

Vivek says
DECEMBER 13, 2015 AT 10:17 PM
Proxy Object.
Reply

Darshita Patel says
OCTOBER 25, 2015 AT 6:06 PM
Question I was asked in an interview : can you drop a table using hibernate/hql? and how ?
I believe it will be good to add this question in the list.
Thanks.
Reply

Himansu Nayak says
AUGUST 21, 2016 AT 10:43 AM
Not possible without Native QL
session.createSQLQuery("Native SQL").executeUpdate();
Reply

DBA says
OCTOBER 4, 2015 AT 11:48 AM
Hi Pankaj,
How can i connect java application to mainframe system using hibernate ?
What configurations need to perform this.
Reply

Himansu Nayak says
AUGUST 21, 2016 AT 10:57 AM
hibernate is not a protocol
Reply

Nagaraju Gunji says
SEPTEMBER 3, 2015 AT 12:57 AM

How to create custom datatypes in Hibernate, and use of filters in Hibernate?
Reply

Nagaraju Gunji says
SEPTEMBER 3, 2015 AT 12:56 AM
Can we maintain two DB configurations in single hibernate configuration file? If yes, How?
If No, Why?
Reply

Sevak says
JANUARY 19, 2016 AT 6:36 AM
No as it won't validate as per hibernate dtd.
Reply

Himansu Nayak says
AUGUST 21, 2016 AT 10:40 AM
If you go with Hibernate-JPA then you can configure multiple persistence-unit in persistence.xml.
Each persistence unit defines a datasource.
Reply

Yegor says
AUGUST 26, 2015 AT 2:17 AM
Short and very helpful
thanks a lot
Reply

Swetha says
JULY 6, 2015 AT 1:10 AM
Very useful for interviews. Great work.
Reply

vishwas says
JUNE 22, 2015 AT 4:06 PM
oh man ur super !!!
Reply

hemant masal says
JUNE 2, 2015 AT 6:00 AM
hi sir
it's very useful for the interview
Reply

mohit rathod says
MAY 26, 2015 AT 4:54 AM
Excellent but
i have to look for
Avoid Many-to-Many relationships, it can be easily implemented using bidirectional One-to-Many and Many-to-One relationships.
Reply

Kapil Bhardwaj says
MAY 23, 2015 AT 8:04 PM
Thanks a lot Pankaj,amazing collections of Questions.It covered all the questions.However,i wanted to know regarding one question:-
Q) How to alter Database specific configuration information in Hibernate without restarting the entire application?
Regards,
Kapil Bhardwaj
Reply

Gagan says
MAY 22, 2015 AT 2:19 AM
Nice article
Reply

Bhaskar says
MAY 3, 2015 AT 8:01 AM
hai sir
This every help-full interview sir
Thanks sir
Bhaskar
Reply

Piyush Beli says
APRIL 28, 2015 AT 11:49 PM
This was really very concise but very well explained article. I have bookmarked this page and I can always refer this page with confidence if I want to prepare for Hibernate interview.
Keep it up !!
Reply

nahar says
APRIL 3, 2015 AT 11:43 AM
really good sir it helping me but sir acctullly you haven't give any example please sir next time provide some example ,
but nice for imporvement
Reply

Swadhi says
APRIL 2, 2015 AT 8:15 PM
really appreciate the effort! could you add questions on Hibernate inheritance with examples? thanks!
Reply

veerendra kumar says
MARCH 26, 2015 AT 10:27 PM
I really like this site for covered all java technologies,frameworks…etc.here hibernate questions is awesome,very thanks pankaj.
Reply

Manula Thantriwatte says
MARCH 17, 2015 AT 3:38 AM
Thank you very much for sharing your knowledge.
Reply

Abhishek says
MARCH 4, 2015 AT 1:09 AM
Thank You Very much for a very helpful set of questions □
Reply

AMit Kumar Gupta says
FEBRUARY 20, 2015 AT 3:20 PM
This collection is really useful. Thanks a lot.
Reply

Gaurav Kumar Chauhan says
FEBRUARY 19, 2015 AT 9:43 PM
Thanks Pankaj, this blog really helps to freshers as well as for experience person.
Good work bro..appreciate your efforts and knowledge.
Keep it up.
Reply

kolem says
FEBRUARY 14, 2015 AT 11:44 AM
Perfect article! I used it for interview preparation, it helped me to recover knowledges and I also learnt several new things. Many thanks!
Reply

Rohan P. says
FEBRUARY 14, 2015 AT 10:55 AM
Hey hi pankaj, great work!!!
I have been following all of your Interview related question and answer, its lot much helpful..

In an recent interview, I have been asked a question “How to create two same tables in different Schema’s by using single configuration file?” And “How to use single configuration file to work with two different databases?”, I was clueless for these question & these questions put me out of the interview. Can you please answer them….
Thank you.
Reply

Pankaj says
FEBRUARY 14, 2015 AT 8:36 PM
You can create multiple session-factory configurations with different name and then get them in the Code by name, such as
session-factory name="OracleDB" and
session-factory name="MySQLDB".
Reply

Raghav says
AUGUST 3, 2015 AT 5:31 PM
Hi Pankaj,
Could you please give one example for “How to create two same tables in different Schema’s by using single configuration file?” And “How to use single configuration file to work with two different databases?” Thanks.
Reply

Himansu Nayak says
AUGUST 21, 2016 AT 11:03 AM
check @Table() with catalog and schema
Reply

nithu says
JANUARY 8, 2015 AT 5:43 AM
Thanks a lot Pankaj, this tutorial is really helpful.. Keep posting !!
Reply

nirbi says
JANUARY 2, 2015 AT 9:35 AM
Sorry for interrupting you ,but i have a doubt related to named queries that u have mentioned above. we are centralizing the query in one location for multi uses in our application instead of scattering it but what are the the cons or draw backs of it???
Reply

Gaurav Kumar says
JANUARY 1, 2015 AT 3:59 AM
I think answer to Question 14 is incorrect. Both get and load supports lazy loading. The only difference is that get returns null if the row is not available corresponding to passed id while load throws an exception.
Reply

Gaurav Kumar says
JANUARY 1, 2015 AT 4:17 AM
I was wrong. I misunderstood the proxy object thing but now its clear. The answer is perfectly correct. Thanks
Reply

Mehatab Syed says
DECEMBER 21, 2014 AT 1:30 PM
Great Job Pankaj. Its Awesome.
Reply

ali says
DECEMBER 1, 2014 AT 9:58 PM
Hi Pankaj
“Hibernate provides reference implementation of Java Persistence API”
As this link shows Eclipselink is the RI of the Java Persistance API not the Hibernate.
please let me know if I’m wrong.
regards
Reply

Pavel Strelchenko says
NOVEMBER 28, 2014 AT 12:12 PM
"Hibernate provide option through which we can create database tables too, for JDBC tables must exist in the database." (question 4)
Sorry, but we also can create tables through JDBC native SQL queries. For example:

```
String createQueryText = "CREATE TABLE `Employee` ( \n" +
"ID int not null primary key\n" +
");";
Statement statement = connection.createStatement();
statement.execute(createQueryText);
```

P.S. Thanks for your article! It's very usefull for me =)
Reply

Pankaj says
NOVEMBER 28, 2014 AT 8:49 PM
Yes we can, but in Hibernate we can configure it to create if table doesn't exist. If you are using JDBC in any application, the code will try to create table every time it's executed and throw exception, yes we can argue that we can catch and ignore it. But it's very clumsy and not a preferred approach.
Reply

Gopal says
NOVEMBER 16, 2014 AT 11:18 PM
Great work Pankaj....
Could you please add N+1 update problem question and explanation.
Thanks
Reply

Dastagiri Reddy says
OCTOBER 27, 2014 AT 11:42 PM
Very nice. Appreciate your efforts.
Reply

Bhavani says
OCTOBER 20, 2014 AT 2:26 AM

Really useful for interviews, Thanks
Reply

Avnish Kumar says
OCTOBER 18, 2014 AT 4:14 AM
Hi,
can you please give a detailed explanation , how to call a procedure in Hibernate 3.0 and struts 2.0 , I am unable to do it ?
Reply

varalakshmi says
FEBRUARY 17, 2015 AT 11:05 PM
Hey we can call procudure using createSQLQuery::
session.createSQLQuery("CALL procedure_name(:param)").setParameter("param", param).executeUpdate()
Reply

Savan Patel says
OCTOBER 17, 2014 AT 2:40 AM
Great Work …
Mostly all the Hibernate Question covered in this post..
#Hibernate
Reply

Deepak says
OCTOBER 16, 2014 AT 11:45 PM
Thanks Pankaj,this is very helpful……
Reply

Ram says
SEPTEMBER 30, 2014 AT 5:59 AM
Good Interview Questions.
Reply

arun says
SEPTEMBER 20, 2014 AT 12:56 AM
very nice questions. cheers.
Reply

rohit says
SEPTEMBER 3, 2014 AT 9:11 PM
Question's covered are very Good.:-)
Reply

Baidyanath Sinha says
SEPTEMBER 3, 2014 AT 12:47 PM
Interview questions and tutorials covered are awesome Pankaj....:)
Reply

Ajit Kumar says
AUGUST 16, 2014 AT 7:47 PM
Excellent work Pankaj. Appreciate your efforts.
Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

HIBERNATE FRAMEWORK

Hibernate Tutorial

- Hibernate Example
- Hibernate SessionFactory
- Hibernate Session get load
- Hibernate Session save
- HQL Example
- Hibernate Criteria
- Hibernate SQL
- Hibernate Named Query
- Hibernate Log4J
- Hibernate Validator
- Hibernate Tomcat DataSource

Hibernate Mapping

- Hibernate One to One Mapping
- Hibernate One to Many Mapping
- Hibernate Many to Many Join Tables

Hibernate Caching

- Hibernate Cache
- Hibernate EHCache

Hibernate Integrations

- Hibernate Spring
- Hibernate Spring MVC
- Hibernate Struts 2
- Hibernate Primefaces
- Hibernate Primefaces Spring
- Hibernate SpringRoo Primefaces
- Hibernate JSF Spring

Miscellaneous

- Hibernate Tools Eclipse Plugin
- Hibernate Configuration Offline
- [Solved] No identifier specified
- Hibernate Program Not Terminating
- Access to DialectResolutionInfo
- get is not valid
- No CurrentSessionContext configured
- Hibernate Interview Questions

IMPORTANT INTERVIEW QUESTIONS

Java Interview Questions

- Core Java Interview Questions
- String Interview Questions
- Multithreading Interview Questions
- Collections Interview Questions
- Exception Interview Questions
- Java Programming Interview Questions
- Java 8 Interview Questions Part 1
- Java 8 Interview Questions Part 2
- Servlet Interview Questions
- JSP Interview Questions
- Struts 2 Interview Questions
- JDBC Interview Questions
- Spring Interview Questions
- Hibernate Interview Questions
- JSF Interview Questions
- Web Services Interview Questions
- Scala Basic Interview Questions
- Scala Intermediate Interview Questions
- Scala Advanced Interview Questions
- Scala Interview Questions Summary
- Common Job Interview Questions

Miscellaneous

- Java ClassLoader
- String StringBuffer StringBuilder
- Java is Pass By Value
- Java Heap vs Stack Memory

RECOMMENDED TUTORIALS

Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial