46 hibernate interview questions and answers

Hibernate interview questions and answers for freshers and experienced candidates. These interview questions and answers on Hibernate will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts. Many candidates appear for the interview for one role - many of the them give the right answers to the questions asked. The one who provides the best answer with a perfect presentation is the one who wins the interview race. The set of Hibernate interview questions here ensures that you offer a perfect answer to the interview questions posed to you.

Hibernate interview questions last updated on May 17, 2011

Part 1 **Part 2** **Part 3** **Part 4** **Part 5**

Hibernate interview questions - posted on March 03, 2014 at 15:05 PM by Nihal Singh

Q:1 What is Hibernate?

Ans: Hibernate is an ORM (Object Relational Mapping) and persistent framework. The framework helps to map plain java object to relational database table using xml configuration file.

The framework helps to perform following things.

• Perform basic CURD operations.
• Write queries referring to java classes (HQL queries).
• Facilities to specify metadata.
• Dirty checking, lazy association fetching.

Q: 2 Why hibernate and how does it help in the programming?

Ans: The main advantage of Hibernate (ORM) framework is that it shields developer to write a messy SQL. Apart from that ORM provides following benefits.

• Improve Productivity of the developer by providing high level object oriented API (e.g. API for easily maintaining the connection to data base, mapping java classes to relational database tables), less java code to write, helps to avoid writing SQL query.
• Improved performance by providing sophisticated caching, lazy loading and eager loading features.
• Provide portability, the framework helps to generate database specific SQL for you.

Q:3 How will you configure Hibernate?

Ans: To configure hibernate, you need hibernate.cfg.xml or hibernate.properties file and *.hbm.xml files, all these files are used by Configuration class to create sessionFactory, which in turn creates the session instances. Session instances are the primary interface for persistence services.

The hibernate.cfg.xml or hibernate.properties files are used to configure the hibernate service (database connection driver class, database connection URL, connection user name, connection password, dialect, mapping resources etc.).

The *hbm.xml files are used for mapping persistent objects to relational database.

From Java 5 onwards you can configure and map persistent objects through annotations.

Q:4 Which settings will be loaded if both hibernate.properties and hibernat.cf.xml files are present in the classpath?

Ans: If both hibernate.properties and hibernate.cfg.xml files are present in the classpath then hibernate.cfg.xml file will override the settings found in hibernate.properties. So please make sure that your project should include either hibernate.properties or hibernate.cfg.xml file.

Q:5 What are the Core interfaces of Hibernate framework?

Ans: There are five core interfaces being used extensively in every Hibernate application. Using these interfaces you can store or retrieve any persistent objects and also control transactions.

- Session interface
- SessionFactory interface
- Configuration interface
- Transaction interface
- Query and Criteria interfaces

Q:6 What is the role of Session interface in Hibernate?

Ans: The session interface is the primary interface used in Hibernate Application. It is single threaded sort-lived object and represents conversation between Application and the persistent store. It helps to create query objects to retrieve persistent objects.

Interview questions
Java interview questions
Java FAQs
Download Java/Oracle FAQ
Test Java skills New
Test JDBC skills New
Test EJB skills New

Java object, class, method
Java operator
Java variables
Java overloading overriding
Java abstract classes
Java data types
Java arrays
Java exception
Java events
Java virtual machine
Java input and output
Java URL connections
Java sockets
JNDI
Java applets
Java AWT
Java drawing AWT components
Core java
JDBC
JSP
EJB
J2EE
JNI
Servlets
Struts
Java threading
J2ME
JMS
Java web services
RMI
Internationalization
JavaScript

You can get the session object from session factory

Session session = sessionFactory.openSession();

Session Interface role:

--Wraps a JDBC connection
--Factory for Transaction
--Holds a mandatory (first-level) cache of persistent objects, used when navigating the object graph or looking up objects by identifier

Q: 7 What is the role of SessionFactory?

Ans: The application obtains session object from SessionFactory interface. Typically there should be only one sessionFacory for whole application and is loaded during application initialization. The SessionFactory caches generate SQL Statement and other mapping metadata that Hibernate use at runtime. It also hold cached data that has been read in one unit of work and can be reused in a future unit of work.

You can get the instance of SessionFactory by the configuration object as below

SessionFactory sessionFactory = configuration.buildSessionFactory();

Q:8 How do you implement one to one relationship in Hibernate with XML mapping?

Ans: For example you have table emp and emp_detail and assuming there is a one to one relationship between them. For the above tables you have to create corresponding POJO classes and hbm.xml files.

So for emp table the java class is Employee.java with property empId and xml file is emp.hbm.xml.

And for emp_details the java class is EmployeeDetail.java (properties are name, address etc.) and xml file is empdetail.hbm.xml.

So the final code will look like as below

```
package com.test.onetoone.mapping

public class Employee implements java.io.Serializable{
private Integer empId;
private EmployeeDetail empDetail;

}
package com.test.onetoone.mapping
public class EmployeeDetail implements java.io.Serializable{
private Integer empId;
private Employee emp;
private String name;
private String address;
}
----- emp.hbm.xml ----
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.test.onetoone.mapping.Employee" table="emp">
<id name="empId" type="java.lang.Integer">
<column name="EMP_ID" />
<generator class="identity" />
</id>
<one-to-one name="empDetail" class="com.test.onetoone.mapping.EmployeeDetail"
cascade="save-update"></one-to-one>
</class>
</hibernate-mapping>

**********empdetails.hbm.xml**********

<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.test.onetoone.mapping.EmployeeDetail" table="emp_detail"
catalog="mkyongdb">
<id name="stockId" type="java.lang.Integer">
<column name="EMP_ID" />
<generator class="foreign">
<param name="property">emp</param>
</generator>
</id>
<one-to-one name="emp" class="com.test.onetoone.mapping.Employee"
constrained="true"></one-to-one>
<property name="name" type="string">
<column name="EMP_NAME" length="100" not-null="true" />
</property>
<property name="address" type="string">
<column name="EMP_ADDR" not-null="true" />
</property>
</class>
</hibernate-mapping>
```

EJB Architecture
Spring
Java Transaction API – JTA
JBOSS
Java transaction services, JTS
Java GUI Framework interview
SWT/JFace interview
MVC and Jface
JavaServer Faces
Hibernate interview
JAAS
JavaMail interview
Tomcat
MIDP interview
Weblogic interview
Ant interview
Websphere interview
Ruby interview
jQuery interview
Aspect Oriented Programming
Java design patterns
Java localization

Q:9 How do you implement one to one relationship in Hibernate with java annotation?

Ans: Taking the same Employee and Employee Details example of the above question.

Employee.java

```
package com.test.onetoone.mapping
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import static javax.persistence.GenerationType.IDENTITY;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;
@Entity
@Table(name = "emp")
public class Employee implements java.io.Serializable{
private Integer empId;
private EmployeeDetail empDetail;
public Employee(){}
@Id
@GeneratedValue(strategy = IDENTITY)
@Column(name = "EMP_ID", unique = true, nullable = false)
public Integer getEmpId(){
return this.empId;
}
public void setEmpId(Integer empId){
this.empId = empId;
}
@OneToOne(fetch = FetchType.LAZY, mappedBy = "emp", cascade = CascadeType.ALL)
public EmployeeDetail getEmpDetail() {
return this.empDetail;
}
public void setEmpDetail(EmployeeDetail empDetail) {
this.empDetail = empDetail; }
}
```

File: EmployeeDetail.java

```
package com.test.onetoone.mapping
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import org.hibernate.annotations.GenericGenerator;
import org.hibernate.annotations.Parameter;
@Entity
@Table(name = "emp_detail")
public class EmployeeDetail implements java.io.Serializable {
private Integer empId;
private Employee emp;
private String name;
private String address;
public EmployeeDetail() {
}
public EmployeeDetail(Employee emp, String name, String address) {
this.emp = emp;
this.name = name;
this.address = address;
}
@GenericGenerator(name = "generator", strategy = "foreign",
parameters = @Parameter(name = "property", value = "emp"))
@Id
@GeneratedValue(generator = "generator")
@Column(name = "EMP_ID", unique = true, nullable = false)
public Integer getEmpId() {
return this.empId;
}
public void setEmpId(Integer empId) {
this.empId = empId;
}
@OneToOne(fetch = FetchType.LAZY)
@PrimaryKeyJoinColumn
public Employee getEmp() {
return this.emp; }

public void setEmp(Employee emp) {
this.emp = emp;
```

```
}
@Column(name = "ADDRESS", nullable = false, length = 400)
public String getAddress() {
return this.address;
}
public void setAddress(String address) {
this.address = address;
}
@Column(name = "EMP_NAME", nullable = false)
public String getName() {
return this.name;
}
public void setName(String name) {
this.name = name;
}
}
```

3. Hibernate Configuration File

Puts annotated classes Employee.java and EmployeeDetail.java in your Hibernate configuration file, and also MySQL connection details.
File : hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/mytestdb</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">password</property>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="show_sql">true</property>
<mapping class="com.test.onetoone.mapping.Employee" />
<mapping class="com.test.onetoone.mapping.EmployeeDetail" />
</session-factory>
</hibernate-configuration>
```

Q:10 How do you implement one to many relationships in Hibernate?

Ans: For one to many relationships we can consider the example Author and Books (i.e. one Author can have written many books).
Below code will help you to understand how you can implement one to many relationship in hibernate.

File: Author.java

```
package com.test.one.to.many;
java.util.Set;
public class Author implements java.io.Serializable{
private Integer authorId;
private String authorName;
private Set<Book> books;
// getter and setter
}
```

File: Book.java

```
package com.test.one.to.many;
public class Book implements java.io.Serializable{
private Author author;
private Integer bookId;
private String bookName;
// getter and setter
}
```

File: Author.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.test.one.to.many.Author" table="author">
<id name="authorId" type="java.lang.Integer">
<column name="AUTHOR_ID" />
<generator class="identity" />
</id>
<property name="authorName" type="string">
<column name="AUTHOR_NAME" length="100" not-null="true" unique="true" />
</property>
<set name="books" table="book" inverse="true" lazy="true" fetch="select">
<key>
<column name="AUTHOR_ID" not-null="true" />
</key>
<one-to-many class="com.test.one.to.many.Book" />
</set>
</class>
```

```
</hibernate-mapping>
```

File: Book.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.test.one.to.many.Book" table="book">
<id name="bookId" type="java.lang.Integer">
<column name="BOOK_ID" />
<generator class="identity" />
</id>
<many-to-one name="author" class="com.test.one.to.many.Author" fetch="select">
<column name="AUTHOR_ID" not-null="true" />
</many-to-one>
<property name="bookName" type="string">
<column name="BOOK_NAME" />
</property>
</class>
</hibernate-mapping>
```

File: hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/mytestdb</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">password</property>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="show_sql">true</property>
<property name="format_sql">true</property>
<mapping resource="com/test/one/to/many/Author.hbm.xml" />
<mapping resource="com/test/one/to/many/Book.hbm.xml" />
</session-factory>
</hibernate-configuration>
```

Q:11 How to implement one to many relationships with Annotation?

Ans: You can implement one to many relationship with the help Annotation also. Below code will help you to understand the one to many relationship with java Annotation.

File : Author.java

```
package com.test.one.to.many;
import java.util.Set;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import static javax.persistence.GenerationType.IDENTITY;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "Author")
public class Author implements java.io.Serializable {
private Integer authorId;
private String authorName;
private Set<Book> books;
// getter and setter

public Author() {
}
@Id
@GeneratedValue(strategy = IDENTITY)
@Column(name = "AUTHOR_ID", unique = true, nullable = false)
public Integer getAuthorId() {
return this.authorId;
}

public void setAuthorId(Integer authorId) {
this.authorId = authorId;
}

@Column(name = "AUTHOR_NAME", nullable = false, length = 100)
public String getAuthorName() {
return this.authorName;
}
```

```
public void setAuthorName(String authorName) {
this.authorName = authorName;
}
@OneToMany(fetch = FetchType.LAZY, mappedBy = "author")
public Set<Book> getBooks() {
return this.books;
}
public void setBooks(Set<Book> books) {
this.books = books;
}

}
```

File : Book.java

```
package com.test.one.to.many;
import java.util.Date;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import static javax.persistence.GenerationType.IDENTITY;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;

@Entity
@Table(name = "book")
public class Book implements java.io.Serializable {
private Author author;
private Integer bookId;
private String bookName;
public Book() {
}
@Id
@GeneratedValue(strategy = IDENTITY)
@Column(name = "BOOK_ID", unique = true, nullable = false)
public Integer getBookId() {
return this.bookId;
}
public void setBookId(Integer bookId) {
this.bookId = bookId;
}
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "AUTHOR_ID", nullable = false)
public Author getAuthor() {
return this.auther;
}
public void setAuther(Author author) {
this.auther = auther;
}
@Column(name = "BOOK_NAME", length = 400, nulaable=false)
public Float getBookName() {
return this.bookName;
}
public void setBookName(String bookName) {
this.bookName = bookName;
}

}
```

3. Hibernate Configuration File

Puts annotated classes Author.java and Book.java in hibernate.cfg.xml like this :

File : hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">


<hibernate-configuration>
<session-factory>
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/mytestdb</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">password</property>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="show_sql">true</property>
<mapping class="com.test.one.to.many.Author" />
<mapping class="com.test.one.to.many.Book" />
</session-factory>
</hibernate-configuration>
```

Q:12 How will you integrate Hibernate with spring framework?

Ans: To integrate hibernate with spring framework, the hibernate configuration will go in spring configuration file.
The configuration file will look like as below

```
<beans>
<bean id="propertyConfigurer"
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"
p:location="/WEB-INF/jdbc.properties"></bean>
<!—jdbc.properties database related properties -?

<bean id="dataSource"
class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close"
p:driverClassName="${jdbc.driverClassName}"
p:url="${jdbc.databaseurl}" p:username="${jdbc.username}"
p:password="${jdbc.password}"></bean>

<bean id="sessionFactory"
class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
<property name="dataSource" ref="dataSource"></property>
<property name="configLocation">
<value>classpath:hibernate.cfg.xml</value>
</property>
<property name="configurationClass">
<value>org.hibernate.cfg.AnnotationConfiguration</value>
</property>
<property name="hibernateProperties">
<props>
<prop key="hibernate.dialect">${jdbc.dialect}</prop>
<prop key="hibernate.show_sql">true</prop>
</props>
</property>
</bean>

<bean id="employeeDAO" class="com.test.dao.EmployeeDaoImpl"></bean>
<bean id="employeeManager" class="com.test.service.EmployeeManagerImpl"></bean>

<tx:annotation-driven />

<bean id="transactionManager"
class="org.springframework.orm.hibernate3.HibernateTransactionManager">
<property name="sessionFactory" ref="sessionFactory"></property>
</bean>
</beans>
```

Q:13 What are the Collection types in Hibernate?

Ans:
- Bag
- Set
- List
- Array
- Map

Q:14 What is HibernateTemplate?

Ans: The spring framework provides HibernateTemplate (org.springframework.orm.hibernate.HibernateTemplate) which is kind of helper class and provides following benefits.

- HibernateTemplate class simplifies interaction with Hibernate session.
- Common functions are simplified to single method calls.
- Sessions are automatically closed.
- Exception are automatically caught and converted to runtime exceptions.

Q:15 What is the difference between load() and get() method?

Ans:

load():

- Use load() method only when you are sure that object you want to read already exists.
- If unique Id of an object does not exists in database then load() method will throw an exception.
- load() method return proxy object default and database won't be hit until the proxy is first invoked.

get():

- Use get() method when you are not sure obout the object existance in the database.
- If object does not exists in the database, the get() method will return null.
- get() method will hit database immediately.

Q:16 What is lazy fetching in Hibernate?

Ans: In Hibernate Lazy fetching is associated with child objects loading for its parents. Through Hibernate mapping file (.hbm.xml) you can specified the selection of loading child objects. By default Hibernate does not load child objects. Lazy=rue means not to load the child objects.

Q:17 What is the difference between merge and update method?

Ans: Use update() method when you are sure that 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.

Q:18 How do you define sequence generated primary key in hibernate?

Ans:
Using <generator> tag.
Example:-
<id column="CUST_ID" name="id" type="java.lang.Long">
<generator class="sequence">
<param name="table">SEQUENCE_NAME</param>
<generator>
</id>

Q:19 What are the different types of caches in Hibernate?

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

Q:20 What do you mean by Named – SQL query?

Ans: Named SQL queries are defined in the mapping xml document and called wherever required.
Example:
<sql-query name = "empdetails">
<return alias="emp" class="com.test.Employee"/>
SELECT emp.EMP_ID AS {emp.empid},
emp.EMP_ADDRESS AS {emp.address},
emp.EMP_NAME AS {emp.name}
FROM Employee EMP WHERE emp.NAME LIKE :name
</sql-query>
Invoke Named Query :
List people = session.getNamedQuery("empdetails")
.setString("Deepak", name)
.setMaxResults(50)
.list();

1.What is ORM?

Latest answer: Object Relational Model is a programming technique. This technique is used to convert the data from an incompatible type to that of a relational database. The mapping is done by using the OOP.....
Read answer

2. What is Hibernate?

Latest answer: Hibernate is an ORM tool. Hibernate is a solution for object-relational mapping and a persistence management layer. For example a java application is used to save data of an object to a database.....
Read answer

3. Why do you need ORM tools like hibernate?

Latest answer: Hibernate is open source ORM tool. Hibernate reduces the time to perform database operations. Hibernate has advantages over using entity beans or JDBC calls. The implementation is done by just using POJOs.....
Read answer

4. What are the main advantages of ORM like hibernate?

Latest answer: The SQL code / statements in the application can be eliminated without writing complex JDBC / Entity Bean code.....
Read answer

5. What are the core interfaces of Hibernate framework?

Latest answer: Session Interface: The basic interface for all hibernate applications. The instances are light weighted and can be created and destroyed without expensive process.....
Read answer

6. Explain how to configure Hibernate.

Latest answer: Hibernate uses a file by name hibernate.cfg.xml. This file creates the connection pool and establishes the required environment. A file named .hbm.xml is used to author mappings.....
Read answer

7. Define HibernateTemplate.

Latest answer: HibernateTemplate is a helper class that is used to simplify the data access code. This class supports automatically converts HibernateExceptions which is a checked exception into.....
Read answer

8. What are the benefits of HibernateTemplate?

Latest answer: HibernateTemplate, which is a Spring Template class, can simplify the interactions with Hibernate Sessions.....
Read answer

9. What is Hibernate proxy?

Latest answer: 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.....
Read answer

10. Explain the types of Hibernate instance states.

Latest answer: The persistent class's instance can be in any one of the three different states. These states are defined with a persistence context. The Hibernate has the following instance states:.....
Read answer

11. Explain the Collection types in Hibernate.

Latest answer: A collection is defined as a one-to-many reference. The simplest collection type in Hibernate is.....
Read answer

12. What is lazy initialization in hibernate?

Latest answer: The delaying the object creation or calculating a value or some process until the first time it is needed. The retrieval of particular information only at the time when the object is accessed.....
Read answer

13. What is lazy fetching in hibernate?

Latest answer: 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.....
Read answer

14. What is the difference between sorted and ordered collection in hibernate?

Latest answer:
Sorted Collection
The sorted collection is a collection that is sorted using the Java collections framework. The sorting is done in the memory of JVM that is running hibernate, soon after reading the data from the database.....
Read answer

15. Explain the difference between transient (i.e. newly instantiated) and detached objects in hibernate.

Latest answer: Transient objects do not have association with the databases and session objects. They are simple objects and not persisted to the database.....
Read answer

16. Explain the advantages and disadvantages of detached objects.

Latest answer: Detached objects passing can be done across layers upto the presentation layer without using Data Transfer Objects.....
Read answer

17. What is Hibernate Query Language (HQL)?

Latest answer: Hibernate Query Language is designed for data management using Hibernate technology. It is completely object oriented and hence has notions like inheritance, polymorphism and abstraction.....
Read answer

18. Explain the general flow of Hibernate communication with RDBMS?

Latest answer: The Hibernate configuration is to be loaded and creation of configuration object is done. The mapping of all hbm files will be performed automatically.....
Read answer

19. Explain the role of Session interface in Hibernate.

Latest answer: Session interface is a single threaded object. The representation of single unit of work with the Java application and the persistence database is done by this object.....
Read answer

20. What is a SessionFactory?

Latest answer: 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.....

Read answer

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

Latest answer: The SessionFactory is used to create Sessions. Each application is having usually only one SessionFactory.....
Read answer

22. Explain the difference between load() and get() in Hibernate.

Latest answer: load()
Use this method if it is sure that the objects exist.

The load() method throws an exception,when the unique id could not found in the database.....
Read answer

23. What is the difference between merge and update?

Latest answer: update () : When the session does not contain an persistent instance with the same identifier, and if it is sure use update for the data persistence in hibernate.....
Read answer

24. What is the advantage of Hibernate over jdbc?

Latest answer: Hibernate code will work well for all databases, for ex: Oracle,MySQL, etc. where as JDBC is database specific.....
Read answer

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

Latest answer: 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.....
Read answer

26. Explain the main difference between Entity Beans and Hibernate.

Latest answer: Entity beans are to be implemented by containers, classes, descriptors. Hibernate is just a tool that quickly persist the object tree to a class hierarchy in a database and without using a single SQL.....
Read answer

27. Explain the difference between hibernate and Spring.

Latest answer: Hibernate is an ORM tool for data persistency. Spring is a framework for enterprise applications.....
Read answer

<<**Previous** **Next**>>

Test your Java skills

Java part 1 (39 questions)
Java part 2 (40 questions)
EJB (20 questions)
JDBC (20 questions)
Applet (20 questions)
Struts (21 questions)
Servlets (20 questions)
Java web services (20 questions)

Write your comment - Share Knowledge and Experience

Discussion Board

Hibernate

Questions are really gr!

Shailesh 03-5-2014 09:15 AM