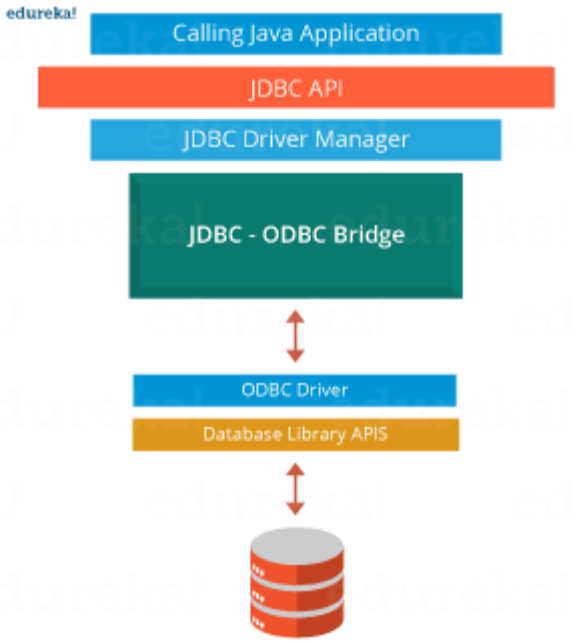
JDBC Interview Questions

1. What is JDBC Driver?

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)



2. What are the steps to connect to a database in java?

- Registering the driver class
- Creating connection
- Creating statement
- Executing queries
- Closing connection

3. What are the JDBC API components?

The java.sql package contains interfaces and classes for JDBC API.

Interfaces:

- Connection

- Statement
- PreparedStatement
- ResultSet
- ResultSetMetaData
- DatabaseMetaData
- CallableStatement etc.

Classes:

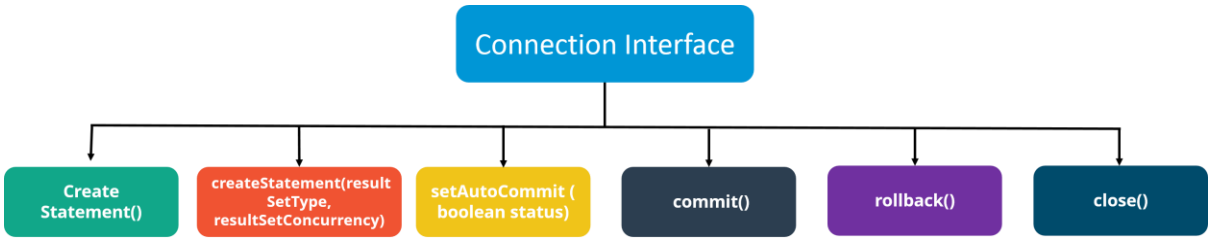
- DriverManager
- Blob
- Clob
- Types
- SQLException etc.

4. What is the role of JDBC DriverManager class?

The DriverManager *class* manages the registered drivers. It can be used to register and unregister drivers. It provides factory method that returns the instance of Connection.

5. What is JDBC Connection interface?

The Connection interface maintains a session with the database. It can be used for transaction management. It provides factory methods that returns the instance of Statement, PreparedStatement, CallableStatement and DatabaseMetaData.



In case you are facing any challenges with these java interview questions, please comment on your problems in the section below.

6. What is the purpose of JDBC ResultSet interface?

The ResultSet object represents a row of a table. It can be used to change the cursor pointer and get the information from the database.

7. What is JDBC ResultSetMetaData interface?

The ResultSetMetaData interface returns the information of table such as total number of columns, column name, column type etc.

8. What is JDBC DatabaseMetaData interface?

The DatabaseMetaData interface returns the information of the database such as username, driver name, driver version, number of tables, number of views etc.

9. What do you mean by batch processing in JDBC?

Batch processing helps you to group related SQL statements into a batch and execute them instead of executing a single query. By using batch processing technique in JDBC, you can execute multiple queries which makes the performance faster.

10. What is the difference between execute, executeQuery, executeUpdate?

Statement ***execute(String query)*** is used to execute any SQL query and it returns TRUE if the result is an ResultSet such as running Select queries. The output is FALSE when there is no ResultSet object such as running Insert or Update queries. We can use *getResultSet()* to get the ResultSet and *getUpdateCount()* method to retrieve the update count.

Statement ***executeQuery(String query)*** is used to execute Select queries and returns the ResultSet. ResultSet returned is never null even if there are no records matching the query. When executing select queries we should use executeQuery method so that if someone tries to execute insert/update statement it will throw java.sql.SQLException with message “executeQuery method can not be used for update”.

Statement **executeUpdate(String query**) is used to execute Insert/Update/Delete (DML) statements or DDL statements that returns nothing. The output is int and equals to the row count for SQL Data Manipulation Language (DML) statements. For DDL statements, the output is 0.

You should use execute() method only when you are not sure about the type of statement else use executeQuery or executeUpdate method.

Q11. What do you understand by JDBC Statements?

JDBC statements are basically the statements which are used to send SQL commands to the database and retrieve data back from the database. Various methods like execute(), executeUpdate(), executeQuery, etc. are provided by JDBC to interact with the database.

JDBC supports 3 types of statements:

1. *Statement:* Used for general purpose access to the database and executes a static SQL query at runtime.
2. *PreparedStatement:* Used to provide input parameters to the query during execution.
3. *CallableStatement:* Used to access the database stored procedures and helps in accepting runtime parameters.

In case you are facing any challenges with these java interview questions, please comment your problems in the section below. Apart from this Java Interview Questions Blog, if you want to get trained from professionals on this technology, you can opt for a structured training from edureka!

Spring Framework – Java Interview Questions

Q1. What is Spring?

Wikipedia defines the Spring framework as “an application framework and inversion of control container for the Java platform. The framework’s core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.” Spring is essentially a lightweight, integrated framework that can be used for developing enterprise applications in java.

Q2. Name the different modules of the Spring framework.

Some of the important Spring Framework modules are:

- Spring Context – for dependency injection.
- Spring AOP – for aspect oriented programming.
- Spring DAO – for database operations using DAO pattern
- Spring JDBC – for JDBC and DataSource support.
- Spring ORM – for ORM tools support such as Hibernate
- Spring Web Module – for creating web applications.
- Spring MVC – Model-View-Controller implementation for creating web applications, web services etc.

Q3. List some of the important annotations in annotation-based Spring configuration.

The important annotations are:

- @Required
- @Autowired
- @Qualifier
- @Resource
- @PostConstruct
- @PreDestroy

Q4. Explain Bean in Spring and List the different Scopes of Spring bean.

Beans are objects that form the backbone of a Spring application. They are managed by the Spring IoC container. In other words, a bean is an object that is instantiated, assembled, and managed by a Spring IoC container.

There are five Scopes defined in Spring beans.

- **Singleton**: Only one instance of the bean will be created for each container. This is the default scope for the spring beans. While using this scope, make sure spring bean doesn't have shared instance variables otherwise it might lead to data inconsistency issues because it's not thread-safe.
- **Prototype**: A new instance will be created every time the bean is requested.
- **Request**: This is same as prototype scope, however it's meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
- **Session**: A new bean will be created for each HTTP session by the container.
- **Global-session**: This is used to create global session beans for Portlet applications.

In case you are facing any challenges with these java interview questions, please comment on your problems in the section below.

Q5. Explain the role of DispatcherServlet and ContextLoaderListener.

DispatcherServlet is basically the front controller in the Spring MVC application as it loads the spring bean configuration file and initializes all the beans that have been configured. If annotations are enabled, it also scans the packages to configure any bean annotated with @Component, @Controller, @Repository or @Service annotations.

ContextLoaderListener, on the other hand, is the listener to start up and shut down the WebApplicationContext in Spring root. Some of its important functions includes tying up the lifecycle of Application Context to the lifecycle of the ServletContext and automating the creation of ApplicationContext.

Q6. What are the differences between constructor injection and setter injection?

No.	Constructor Injection	Setter Injection
1)	No Partial Injection	Partial Injection
2)	Doesn't override the setter property	Overrides the constructor property if both ar
3)	Creates a new instance if any modification occurs	Doesn't create a new instance if you change property value

4)	Better for too many properties	Better for a few properties.

Q7. What is autowiring in Spring? What are the autowiring modes?

Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic. Let's see the code to inject bean using dependency injection.

1. <bean id="emp" class="com.javatpoint.Employee" autowire="byName" />

The autowiring modes are given below:

No.	Mode	Description
1)	no	this is the default mode, it means autowiring is not enabled.
2)	byName	Injects the bean based on the property name. It uses setter method.
3)	byType	Injects the bean based on the property type. It uses setter method.
4)	constructor	It injects the bean using constructor

Q8. How to handle exceptions in Spring MVC Framework?

Spring MVC Framework provides the following ways to help us achieving robust exception handling.

Controller Based:

We can define exception handler methods in our controller classes. All we need is to annotate these methods with @ExceptionHandler annotation.

Global Exception Handler:

Exception Handling is a cross-cutting concern and Spring provides @ControllerAdvice annotation that we can use with any class to define our global exception handler.

HandlerExceptionResolver implementation:

For generic exceptions, most of the times we serve static pages. Spring Framework provides HandlerExceptionResolver interface that we can implement to create global exception handler. The reason behind this additional way to define global exception handler is that Spring framework also provides default implementation classes that we can define in our spring bean configuration file to get spring framework exception handling benefits.

Q9. What are some of the important Spring annotations which you have used?

Some of the Spring annotations that I have used in my project are:

@**Controller** – for controller classes in Spring MVC project.

@**RequestMapping** – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through Spring MVC RequestMapping Annotation Examples

@**ResponseBody** – for sending Object as response, usually for sending XML or JSON data as response.

@**PathVariable** – for mapping dynamic values from the URI to handler method arguments.

@**Autowired** – for autowiring dependencies in spring beans.

@**Qualifier** – with @Autowired annotation to avoid confusion when multiple instances of bean type is present.

@**Service** – for service classes.

@**Scope** – for configuring the scope of the spring bean.

@**Configuration, @ComponentScan and @Bean** – for java based configurations.

AspectJ annotations for configuring aspects and advices , @Aspect, @Before, @After, @Around, @Pointcut, etc.

Q10. How to integrate Spring and Hibernate Frameworks?

We can use Spring ORM module to integrate Spring and Hibernate frameworks if you are using Hibernate 3+ where SessionFactory provides current session, then you should avoid using HibernateTemplate or HibernateDaoSupport classes and better to use DAO pattern with dependency injection for the integration.

Also, Spring ORM provides support for using Spring declarative transaction management, so you should utilize that rather than going for hibernate boiler-plate code for transaction management.

Q11. Name the types of transaction management that Spring supports.

Two types of transaction management are supported by Spring. They are:

1. **Programmatic transaction management:** In this, the transaction is managed with the help of programming. It provides you extreme flexibility, but it is very difficult to maintain.
2. **Declarative transaction management:** In this, transaction management is separated from the business code. Only annotations or XML based configurations are used to manage the transactions.

Apart from these Core Java interview questions for experienced professionals, if you want to get trained by professionals on this technology, you can opt for a structured training from edureka!

Hibernate – Java Interview Questions for Experienced Professionals

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 a framework for mapping application domain objects to the relational database tables and vice versa.

Hibernate provides a reference implementation of Java Persistence API, that makes it a great choice as ORM tool with benefits of loose coupling. We can use the 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.

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

Some of the important benefits of using hibernate framework are:

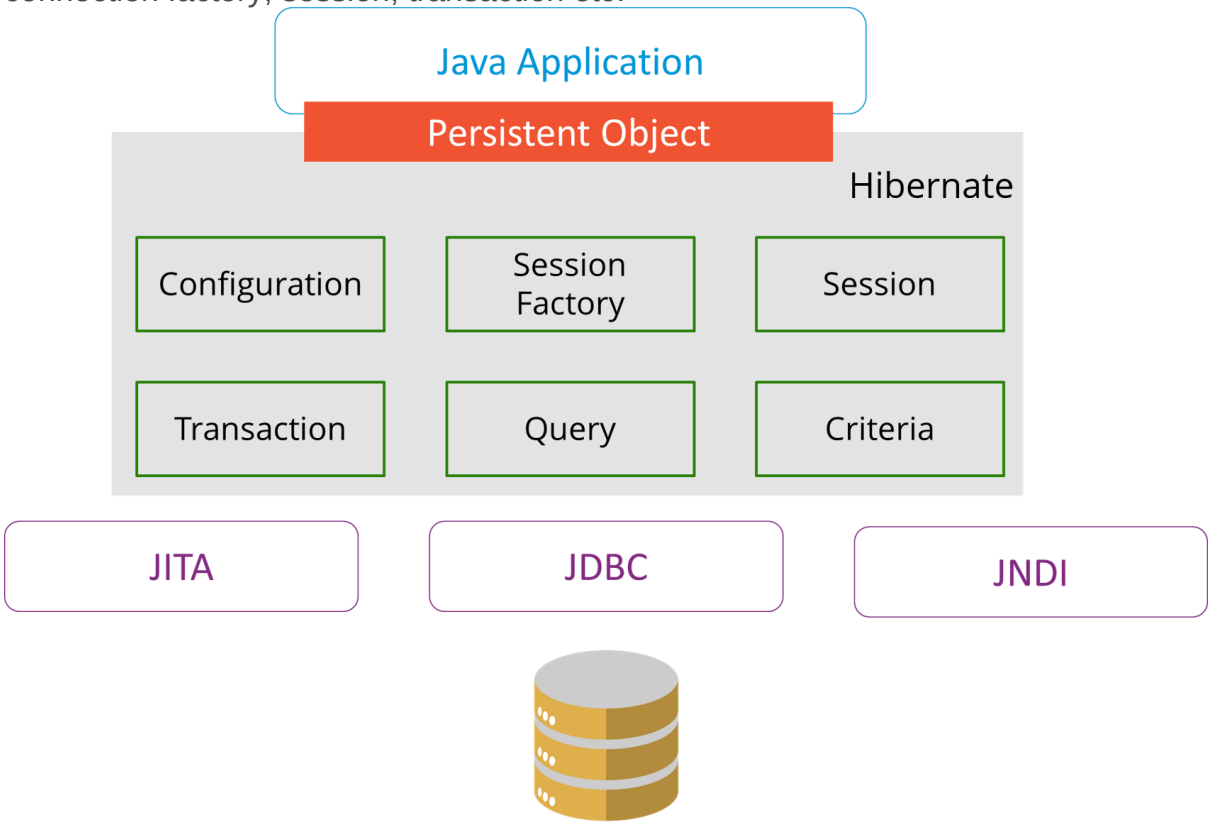
1. Hibernate eliminates all the boiler-plate code that comes with JDBC and takes care of managing resources, so we can focus on business logic.
2. Hibernate framework provides support for XML as well as JPA annotations, that makes our code implementation independent.
3. Hibernate provides a powerful 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 documentation 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.

3. Explain Hibernate architecture.

Hibernate has a layered architecture which helps the user to operate without having to know the underlying APIs. Hibernate makes use of the database and configuration data to provide persistence services (and persistent objects) to the application. It includes many objects such as persistent object, session factory, transaction factory,

connection factory, session, transaction etc.



The Hibernate architecture is categorized in four layers.

- Java application layer
- Hibernate framework layer
- Backhand API layer
- Database layer

4. What are the differences between get and load methods?

The differences between get() and load() methods are given below.

No.	get()	load()
1)	Returns null if object is not found.	Throws ObjectNotFoundException if an obj found.
2)	get() method always hit the database.	load() method doesn't hit the database.
3)	It returns a real object, not a proxy.	It returns a proxy object.

4)	It should be used if you are not sure about the existence of instance.	It should be used if you are sure that the ins exists.

5. 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 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.
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 provides 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 the connection pool. This is a very important feature in enterprise application and completely missing in JDBC API.
9. Hibernate supports JPA annotations, so the code is independent of the implementation and easily replaceable with other ORM tools. JDBC code is very tightly coupled with the application.