Hibernate And Spring

Q.1 What is ORM in Hibernate?

Ans:-

ORM (Object-Relational Mapping) :-

- ORM, an abbreviation for Object-relational mapping, is a programming approach that connects object code to a relational database via a metadata descriptor.
- This object code is developed in object-oriented programming (OOP) languages like Java, Python, C++, C#, etc.
- ORM transforms data between type systems that don't get along in relational databases or OOP languages.
- In other words, it is a technique for storing, recovering, updating, as well as deleting from an object-oriented program in a relational (table) database. Now let's first understand the term object code.

What is Object Code?

Object code is defined as low-level code that is comprehensible by computers. It is generated by the compiler using the source code. In other words, it is a file generated by the compiler containing the instructions for the machine in the form of binary digits.

Advantages of ORM:

- Resolves object code and relational mismatch
- Using ORM, the development process is quite simplified as it automates object to table and table to object conversion which results in lower development and maintenance cost
- The code is less as compared to embedded SQL
- Gives an optimized solution that results in faster application and easier maintenance.

Q.2 What are the advantages of Hibernate over JDBC?

Ans:-

The problems of JDBC API are as follows:

- We need to write a lot of code before and after executing the query, such as creating connection, statement, closing resultset, connection etc.
- We need to perform exception handling code on the database logic.
- We need to handle transaction.
- Repetition of all these codes from one to another database logic is a time consuming task.
- Spring JdbcTemplate eliminates all the above mentioned problems of JDBC API.
- It provides you methods to write the queries directly, so it saves a lot of work and time.

The benefits of HibernateTemplate are:

- HibernateTemplate, which is a Spring Template class, can simplify the interactions with Hibernate Sessions.
- Various common functions are simplified into single method invocations.
- The sessions of hibernate are closed automatically
- The exceptions will be caught automatically, and converts them into runtime exceptions.

- Hibernate supports relationships like One-To-Many,One-To-One, Many-To-Many-to-Many, Many-To-One
- This will also supports collections like List,Set,Map (Only new collections)
- Hibernate supports Inheritance, Associations, Collections
- In hibernate if we save the derived class object, then its base class object will also be stored into
- the database, it means hibernate supporting inheritance
- Hibernate has its own query language, i.e hibernate query language which is database independent So if we change the database, then also our application will works as HQL is database independent
- HQL contains database independent commands
- Hibernate supports annotations, apart from XML

Q.3 What are some of the important interfaces of Hibernate framework?

Ans :-

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

Q.4 What is a Session in Hibernate?

Ans :-

- A Session is used to get a physical connection with a database.
- The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database.
- Persistent objects are saved and retrieved through a Session object.
- The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed.
- The main function of the Session is to offer, create, read, and delete operations for instances of mapped entity classes.
- Instances may exist in one of the following three states at a given point in time −

1. **transient** − A new instance of a persistent class, which is not associated with a Session and has no representation in the database and no identifier value is considered transient by Hibernate.
2. **persistent** − You can make a transient instance persistent by associating it with a Session. A persistent instance has a representation in the database, an identifier value and is associated with a Session.
3. **detached** − Once we close the Hibernate Session, the persistent instance will become a detached instance.

Q.5 What is a SessionFactory?

Ans :-

- SessionFactory is an Interface which is present in org.hibernate package and it is used to create Session Object.
- It is immutable and thread-safe in nature.

Q.6 Can you explain what is lazy loading in hibernate?

Ans :-

- Hibernate Lazy Loading is a popular tool of ORM used by developers of JAVA.
- Hibernate and JPA work along and manages the entity relations.
- The crucial optimization of database technique is hibernated lazy loading relation, The queries are lessened in the database due to this.
- The loading of an entity is done only the first time when you access the entity is called Lazy loading.
- It saves the pre-filling and preloading costs of all the entities in a huge dataset before while you don't need them later.

Q.6 Mention some important annotations used for Hibernate mapping?

Ans :-

Annotations	Use of annotations
@Entity	Used for declaring any POJO class as an entity for a database
@Table	Used to change table details, some of the attributes are- • name – override the table name • schema • catalogue • enforce unique contrants
@Id	Used for declaring a primary key inside our POJO class
@GeneratedValue	Hibernate automatically generate the values with reference to the internal sequence and we don't need to set the values manually.
@Column	It is used to specify column mappings. It means if in case we don't need the name of the column that we declare in POJO but we need to refer to that entity you can change the name for the database table. Some attributes are- • Name – We can change the name of the entity for the database • length – the size of the column mostly used in strings • unique – the column is marked for containing only unique values • nullable – The column values should not be null. It's marked as NOT
@Transient	Tells the hibernate, not to add this particular column
@Temporal	This annotation is used to format the date for storing in the database
@Lob	Used to tell hibernate that it's a large object and is not a simple object
@OrderBy	This annotation will tell hibernate to OrderBy as we do in SQL. For example – we need to order by student first name in ascending order @OrderBy("firstName asc")

Q. What is spring?

Ans:-

- The Spring Framework (Spring) is an open-source application framework that provides infrastructure support for developing Java applications.
- Spring is the most popular application development framework for enterprise Java.
- Spring Framework used to create high performing, easily testable, and reusable code.
- Spring framework is an open source Java platform.
- It was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003.
- Spring is lightweight when it comes to size and transparency.

Q. What is di?

Ans:-

- Dependency Injection is the main functionality provided by Spring IOC(Inversion of Control).
- The Spring-Core module is responsible for injecting dependencies through either Constructor or Setter methods.
- The design principle of Inversion of Control emphasizes keeping the Java classes independent of each other and the container frees them from object creation and maintenance.
- These classes, managed by Spring, must adhere to the standard definition of Java-Bean.
- Dependency Injection in Spring also ensures loose-coupling between the classes.

Q. What is ioc?

Ans:-

- Spring IoC Container is the core of Spring Framework.
- It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection(DI) to manage the components that make up the application.
- It gets the information about the objects from a configuration file(XML) or Java Code or Java Annotations and Java POJO class.
- These objects are called Beans. Since the Controlling of Java objects and their lifecycle is not done by the developers, hence the name **Inversion Of Control**.

Q. Features of spring

Ans:-

1) **IoC container :** Refers to the core container that uses the DI or IoC pattern to implicitly provide an object reference in a class during runtime. This pattern acts as an alternative to the service locator pattern. The IoC container contains assembler code that handles the configuration management of application objects.
   The Spring framework provides two packages, namely org.springframework.beans and org.springframework.context which helps in providing the functionality of the IoC container.

2) **Data access framework:** Allows the developers to use persistence APIs, such as JDBC and Hibernate, for storing persistence data in database. It helps in solving various problems of the developer, such as how to interact with a database connection, how to make sure that the connection is closed, how to deal with exceptions, and how to implement transaction management It also enables the developers to easily write code to access the persistence data throughout the application

3) **Spring MVC framework:** Allows you to build Web applications based on MVC architecture. All the requests made by a user first go through the controller and are then dispatched to different views, that is, to different JSP pages or Servlets. The form handling and form validating features of the Spring MVC framework can be easily integrated with all popular view technologies such as ISP, Jasper Report, FreeMarker, and Velocity.

4) **Transaction management:** Helps in handling transaction management of an application without affecting its code. This framework provides Java Transaction API (JTA) for global transactions managed by an application server and local transactions managed by using the JDBC Hibernate, Java Data Objects (JDO), or other data access APIs. It enables the developer to model a wide range of transactions on the basis of Spring's declarative and programmatic transaction management.

5) **Spring Web Service:** Generates Web service endpoints and definitions based on Java classes, but it is difficult to manage them in an application. To solve this problem, Spring Web Service provides layered-based approaches that are separately managed by Extensible Markup Language (XML) parsing (the technique of reading and manipulating XML). Spring provides effective mapping for transmitting incoming XML message request to an object and the developer to easily distribute XML message (object) between two machines.

6) **JDBC abstraction layer:** Helps the users in handling errors in an easy and efficient manner. The JDBC programming code can be reduced when this abstraction layer is implemented in a Web application. This layer handles exceptions such as DriverNotFound. All SQLExceptions are translated into the DataAccessException class. Spring's data access exception is not JDBC specific and hence Data Access Objects (DAO) are not bound to JDBC only.

7) **Spring TestContext framework:** Provides facilities of unit and integration testing for the Spring applications. Moreover, the Spring TestContext framework provides specific integration testing functionalities such as context management and caching DI of test fixtures, and transactional test management with default rollback semantics.