1. What is Spring Boot?

Ans:- Spring Boot is a Java framework that makes it easier to create and run Java applications. It is a combination of Spring Framework and Embedded Servers. It simplifies the configuration and setup process, allowing developers to focus more on writing code for their applications.

Spring Boot, a module of the spring framework, facilitates Rapid Application Development (RAD) capabilities.

1. What is Dependency Injection?

Ans:- In simple terms, Dependency Injection (DI) in Spring Boot is a design pattern that helps manage the relationships between different components in a software application. It is a way to achieve loose coupling between classes by letting an external entity, typically the Spring framework, handle the creation and injection of dependencies.

1. What is Inversion of Control?

Ans:- Inversion of Control (IoC) is a design principle where the control flow of a program is inverted, meaning the control is transferred from the application code to an external framework. In the context of Spring Boot, IoC is a key concept, and it's often referred to as the IoC container. Dependency injection is one of the way to achive inversion of control in spring boot.

1. Spring Initializer:- By using this we can initialize an empty spring boot project

<Url:-> <https://start.spring.io/>

1. ORM:- ORM is Object Relational Mapping:- **Spring-ORM** is a technique or a Design Pattern used to access a relational database from an object-oriented language. ORM (Object Relation Mapping) covers many persistence technologies.
2. JPA:- ( **Java Persistence API):**It is mainly used to persist data between Java objects and relational databases. It acts as a bridge between object-oriented domain models and relational database systems. It internally use Hibernate
3. Hibernate – It is a Java framework that simplifies the development of Java applications to interact with the database.
4. Hibernate is a non-invasive framework, means it won’t force the programmers to extend /implement any class/interface
5. DAO- Data Access Objet
6. Layers:- Service, Entities, Dao,main,Controller
7. Api-> Application Programming Interface:- It is a set of rules that allow programs to talk to each other. The developer creates the Api on the server and allows the client to talk to it
8. REST:- Representational State Transfer
9. The important HTTP methods are

GET->It reads a resource[@GetMapping(“/ ”)]

PUT->It updates a resource[@PutMapping(“/ ”)]

POST-> It creates a new Resource[@PostMapping(“/ ”)]

DELETE->It deletes a resource[@DeleteMapping(“/ ”)]

1. HQL->Database Independent

From Student (Entity name)

1. @Modifiying used to get know server it is a Delete,Update Query and Must use @Transational in service layer when the method get executed
2. SQL->Database Dependent

Select \* from Student (Table name)

1. @Column- To change the column in the associated table in database
2. @Transient- This tells hibernet not to save a field in database
3. @Temporal- @Temporal over a date field tells hibernet the format in which date needs to be saved
4. @Lob- It tells hibernet that it is a large object, not a simple object
5. @GeneratedValue(strategy=GenerationType.IDENTITY)- For Auto n+1 increment like 1,2,3,4,5,…..,n
6. @Embedebale ->To embade one table in another and have to use getter setter for this in the embded class
7. @Onetoone->It Basically works like foreign Key, In this context @JoinColum(name=””) used to rename joined cloum
8. @OneToOne(cascade = CascadeType.ALL) **PERSIST**: The **persist** operation is used to make a transient instance (e.g., a new **Question** with a new **Answer**) persistent. If you save a **Question** entity, JPA will also save the associated **Answer** entity.
9. @OneToMany->*@OneToMany*(targetEntity = Answer.class,cascade = *CascadeType*.***ALL***)

*@JoinColumn*(name = "question\_id",referencedColumnName = "qid")

private List<Answer> answer;

1. @ManyToMany-> *@ManyToMany*(targetEntity = Employee.class)

private List<Employee>employees; You have write in both classes

1. Fetch Type->

Lazy->Loads data only when the getter or size method called.

Eager->Loads data on the spot