VINAY LAB-14

1.What is Dependency Injection (DI)?

Dependency Injection (DI) is a design pattern and a software development technique used in object-oriented programming to manage the dependencies between different components or objects in a system. It is primarily employed in the context of inversion of control (IoC) containers and frameworks to facilitate loose coupling and improve the maintainability, testability, and flexibility of software systems.

In a typical software application, objects or components often rely on other objects or services to perform their tasks. These dependencies can include things like database connections, file access, network communication, or other custom services. Managing these dependencies manually can lead to tightly coupled code, making it hard to change or test individual components in isolation.

Dependency Injection addresses this issue by allowing dependencies to be injected into an object rather than the object creating or managing them itself.

Here's how it works:

Dependency: A dependency is an object or service that another object needs to perform its function. For example, a UserService object might depend on a UserRepository to retrieve user data from a database.

Dependency Provider: Instead of creating the dependency within the dependent object, the dependency is provided to the dependent object from the outside. This is typically done through constructor injection, method injection, or property injection.

Constructor Injection: Dependencies are passed to the dependent object through its constructor. This is the most common and preferred way of implementing DI. Method Injection: Dependencies are provided to specific methods of the dependent object when needed.

Property Injection: Dependencies are set as properties of the dependent object. IoC Container: In larger applications, an IoC container (Inversion of Control container) is often used to manage the creation and lifetime of objects and their dependencies. The container maintains a registry of object types and their associated dependencies, and it can automatically resolve and inject these dependencies when creating objects.

Benefits of Dependency Injection:

Loose Coupling: DI promotes loose coupling between components, making it easier to change or replace dependencies without affecting the dependent objects.

Testability: By injecting mock or test implementations of dependencies, it becomes easier to unit test individual components in isolation.

Flexibility: Components can be configured and wired together differently at runtime, making it easier to adapt the system to changing requirements.

Reusability: Components can be reused in different contexts or applications because their dependencies can be easily adjusted. Dependency Injection is a powerful technique for managing dependencies in software applications, promoting modularity, maintainability, and testability while reducing the tight coupling between components.

2. What is the purpose of the @Autowired annotation in Spring Boot?

3. Explain the concept of Qualifiers in Spring Boot.

4. What are the different ways to perform Dependency Injection in Spring Boot?

5. Create a SpringBoot application with MVC using Thymeleaf.

(create a form to read a number and check the given number is even or not)