1. Core principles of the Spring framework:
   - Dependency Injection: Objects are given their dependencies rather than creating them.
   - Aspect-oriented programming (AOP): Separates cross-cutting concerns.
   - Loose coupling: Components are independent and can be easily replaced.
   - Interface-driven development: Emphasizes abstraction and interfaces for flexibility.

2. Core annotations in Spring:
   - @Component: Marks a class as a Spring component.
   - @Repository: Specialized component for database operations.
   - @Service: Indicates a service class in Spring.
   - @Controller: Marks a class as a Spring MVC controller.

3. Bean annotations in Spring:
   - @Bean: Used in Java config to declare a bean.
   - @Configuration: Marks a class as providing bean definitions.
   - @ComponentScan: Scans specified packages for Spring components.

4. Dependency Injection (DI):
   - Pattern where a component's dependencies are supplied externally.
   - Promotes decoupling and easier testing.
   - Constructor injection: Dependencies provided via constructor parameters.
   - Setter injection: Dependencies set through setter methods.

5. @Autowired annotation:
   - Automatically wires dependencies into Spring beans.
   - Eliminates the need for explicit bean configuration.

6. Spring bean:
   - Object managed by the Spring IoC container.
   - Configured by the Spring application context.

7. Spring Components:
   - Java classes managed by Spring IoC container.
   - Annotated with @Component, @Service, @Repository, or @Controller.

8. Inversion of Control (IoC):
   - Principle where control over object creation is delegated to an external entity (IoC container).
   - Lifecycle managed by the IoC container.

9. Spring bean scopes:
   - Singleton: Single instance per Spring IoC container.
   - Prototype: New instance each time requested.
   - Request: One instance per HTTP request.
   - Session: One instance per HTTP session.

10. POJO class:
    - Plain Old Java Object.
    - Simple Java class with no special restrictions.
    - Used to represent data and behavior in Java applications.

11. DAO class (Data Access Object):
    - Design pattern to isolate data access code.
    - Provides an interface to interact with the database.
    - Enhances maintainability and testability of code.

- What is **MAVEN**?
- Maven is a powerful build automation tool used primarily for Java projects.
    - What are maven goals?

        Maven goals include compiling source code, running tests, packaging compiled code into JAR or WAR files, deploying artifacts, generating documentation, and managing dependencies.

    - Why do we use Maven?

        For building process of the Java projects.

    - How does maven help in dependency management?

        Developers can use to pom.xml file using the <dependency> to add the dependency to the existing java project.

    - What is the Maven central repository?

        Central repo to store and retrieve the project dependencies

    - How does maven work internally or behind the scenes?

        When maven command is executed, the maven provides the building process of the JAVA project, resolve the dependencies and provide the artifacts.

- What is **Gradle**?
    - Gradle is a powerful build automation tool used for building, testing, and deploying software projects.
    - It uses a Groovy-based domain-specific language (DSL) or Kotlin DSL for writing build scripts, providing flexibility and expressiveness.

        - What are advantages of using gradle?

            Flexibility

DSL support

Higher performance than Maven

- Create a simple application through spring initiliazr using gradle.

- What are the different types of Spring Modules?
  - Spring Core
  - Spring Beans
  - Spring JPA
  - Spring Web
  - Spring Test
  - Spring Security

- MVC Architecture

- What is the flow of requests in MVC architecture?

- What is the significance of the Model, View, and Controller

Model: Represents the application's data and business logic

View: Represents the presentation layer of the application. It is responsible for rendering data to the user in a human-readable format, such as HTML, XML, or JSON.

Controller: Acts as an intermediary between the model and the view. It handles user requests, processes input data, invokes appropriate actions in the model.

Create a simple application using MVC to print hello world! through jsp page:-

Spring boot Assignment - 2

- What is REST?
  - Representational State Transfer uses HTTP requests to perform CRUD operations.
- What is the difference between REST and SOAP?
  - SOAP uses XML messaging
- What are various HTTP methods in REST?
  - GET,POST,PUT,DELETE,PATCH,HEAD,OPTIONS
- What is different between POST and PUT?
  - POST is used for creating a new resource.
  - PUT is used for the updating the existing resource.

- Can we do both operations read and write through one HTTP method?
    - Yes it is possible
- What is the difference between PUT and PATCH?

PATCH is used to update the resource partially