**What is the Spring Framework and its core features?**

The **Spring Framework** is a comprehensive Java framework that simplifies enterprise application development. Its core features include:

1. **Inversion of Control (IoC) & Dependency Injection (DI):** Manages object creation and dependency injection for loose coupling.
2. **Aspect-Oriented Programming (AOP):** Separates cross-cutting concerns like logging and security from business logic.
3. **Transaction Management:** Simplifies managing database transactions.
4. **Spring MVC:** A framework for building web applications using the Model-View-Controller pattern.

What is Spring IoC container?

The **Spring IoC (Inversion of Control) container** is a core concept in the Spring Framework. It manages the lifecycle and configuration of objects in a Spring-based application. The container uses **dependency injection (DI)** to supply objects that an application needs, reducing the coupling between components and promoting loose coupling and better testability.

What is a Spring Bean and what are the different scopes of Spring Beans?

In Spring Framework, a **Spring Bean** is an object that is managed by the Spring **IoC (Inversion of Control) container**. Spring beans are instantiated, configured, and assembled by the Spring container based on the configuration provided in XML files, annotations, or Java configuration classes. The primary purpose of a Spring Bean is to promote **loose coupling** between objects, which helps to make the application more modular and maintainable.

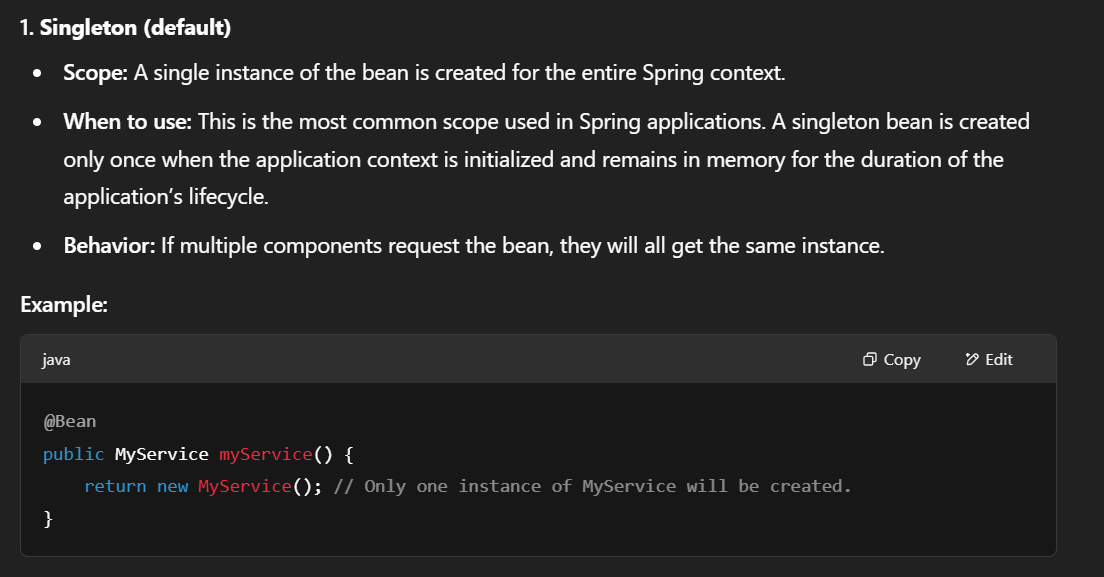
A Spring Bean can be any object in a Spring application, such as a service, a repository, or a controller. These beans can be created automatically by the Spring container based on the configuration and can interact with each other to fulfill the application’s business logic.

A Spring Bean can be defined in several ways:

1. **XML Configuration** (traditional way).
2. **Java-based Configuration** using annotations like @Bean and @Configuration.
3. **Component Scanning** using annotations like @Component, @Service, @Repository, and @Controller.

**Different Scopes of Spring Beans**

Spring provides various scopes for beans, which determine the lifecycle and visibility of beans within the context of the Spring container. The available bean scopes are:



A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

**Dependency Injection (DI)**

A core concept in Spring that allows you to inject dependencies into Spring beans rather than creating them manually within the bean. Spring supports three main types of dependency injection: **Constructor Injection**, **Setter Injection**, and **Field Injection**. Each of these has its use cases and advantages.

**1. Constructor Injection**

In Constructor Injection, dependencies are provided to a bean through its constructor.

**How it works:**

* Spring calls the constructor of the bean and passes the required dependencies as arguments.
* This method is preferred when dependencies are **mandatory** and must be provided at the time of bean creation.

A screenshot of a computer program

AI-generated content may be incorrect.

**2. Setter Injection**

In Setter Injection, dependencies are provided through setter methods after the bean is constructed.

**How it works:**

* Spring creates the bean first and then calls setter methods to inject the dependencies.
* This is useful for **optional dependencies** or when you need to modify the dependencies after bean creation.

A screen shot of a computer program

AI-generated content may be incorrect.

**3. Field Injection**

Field Injection allows Spring to inject dependencies directly into fields, without requiring setter or constructor methods.

**How it works:**

* Spring uses **reflection** to directly inject dependencies into the fields.
* Dependencies are marked with @Autowired (or @Inject), and Spring automatically injects the required dependencies.

A screen shot of a computer program

AI-generated content may be incorrect.

A black and white screen with white text

AI-generated content may be incorrect.