What is IoC (Inversion of Control)?

Inversion of Control (IoC) is a **design principle** used in software development where the control of creating and managing objects is transferred from the application code to a container or framework. Instead of objects controlling their own lifecycle and dependencies, an external container (like Spring's IoC container) manages them.

Key Concept of IoC

- In traditional programming, the **application code** is responsible for creating objects and managing their dependencies using new keyword or direct instantiation.
- With IoC, this responsibility is **inverted** and given to a framework (**like Spring**). The **Spring IoC container** creates objects, sets up their dependencies, and manages their lifecycle.

How IoC Works: Dependency Injection (DI)

In Spring, IoC is primarily implemented using **Dependency Injection (DI)**. DI is the process where the Spring IoC container injects the necessary dependencies into objects automatically.

Types of Dependency Injection in Spring:

1. **Constructor Injection**: Dependencies are provided as constructor arguments.
2. **Setter Injection**: Dependencies are provided using setter methods.
3. **Field Injection**: Dependencies are injected directly into fields using annotations (e.g., @Autowired).

- **Note: ApplicationContext** is **not IoC itself**, but it is the **implementation of the IoC container** in Spring.

- It embodies the **Inversion of Control** principle by taking over the responsibility of creating, configuring, and managing the lifecycle

of beans, allowing developers to focus on business logic rather than object management.

Thus, **ApplicationContext** is a key component that enables Spring's IoC capabilities, making it a cornerstone of the Spring Framework

---