Dependency Injection in Spring Boot - Cheat Sheet

Definition (Simple Words)

Dependency Injection is a design pattern in Spring Boot where the framework automatically provides the required objects (dependencies) to a class instead of the class creating them manually. Spring uses the IoC container to manage and inject these dependencies.

IoC Container Explanation

The IoC (Inversion of Control) container in Spring manages object creation, lifecycle, and dependency injection. It creates beans and injects them into classes using annotations like @Component, @Service, and @Autowired.

Types of Dependency Injection

• Constructor Injection – Dependencies are injected through constructor (Recommended)

• Setter Injection – Dependencies injected through setter methods

• Field Injection – Dependencies injected directly into fields (Not recommended)

Constructor vs Setter vs Field Injection

Type	Best For	Recommended
Constructor Injection	Required dependencies	Yes
Setter Injection	Optional dependencies	Sometimes
Field Injection	Quick testing only	No

Spring Boot Code Example

```
@Service
public class UserService {
private final UserRepository userRepository;

public UserService(UserRepository userRepository) {
this.userRepository = userRepository;
}
}
```

Common Mistakes

• Using field injection in production

• Creating objects manually with new keyword
• Not using @Service or @Repository annotations
• Circular dependency between beans

Interview Questions

• What is dependency injection in Spring Boot?
• What is IoC container?
• Constructor vs field injection?
• Why constructor injection is recommended?
• What is @Autowired?

Best Practices (2026)

• Use constructor injection by default
• Avoid field injection
• Use interfaces for services
• Keep dependencies minimal
• Write unit tests with DI