

Spring Core Made Easy



What is Spring ?

- Lightweight, powerful framework for Java
- Helps build scalable, maintainable enterprise apps
- Modules: Core, MVC, Security, Boot, Data JPA, etc.
- Today's focus: **Spring Core**



☞ **Think of it as the foundation or "heart" of Spring!**



Framework?

- built on top of the programming
- pre-built structure or set of tools to develop your application
- has libraries, templates and guidelines to streamline your development process
- Ex:- Django, Angular, Ruby on Rails



Why Do We Need Spring Core?

- Reduces tight coupling between classes
- Promotes reusable, testable code
- Manages object creation and wiring (IoC)
- Enables **Dependency Injection** (DI)



Example: In a bank, managers don't hire directly — HR does!



What Does Spring Core Include?

- **Beans** – Objects managed by Spring
- **IoC Container** – Manages beans and their dependencies
- **Dependency Injection** – Injects required objects
- **ApplicationContext** – Central interface for accessing beans



Real-Life Analogy – Banking

Java Concept

- BankService
- AccountRepository
- Spring Container
- DI

Banking Analogy

- Bank Manager
- Cashier
- HR Department
- HR assigns right employees

Spring manages everything for you, just like HR handles hiring



What is a Bean?

- A Java object created and managed by the Spring container
- Defined in configuration files or using annotations

```
@Component
```

```
public class AccountService  
{  
  
    // business logic  
  
}
```

Tip: Add `@Component`, and Spring handles the rest!



What is a Dependency Injection?

- DI allows Spring to **inject required objects automatically**
- Promotes loose coupling

```
@Service  
  
public class BankService {  
  
    @Autowired  
  
    private AccountService  
    accountService;  
  
}
```

☞ Spring says: “Don’t create objects yourself, I’ll do it



Types of DI in Spring

- **Constructor Injection**
- **Setter Injection**
- **Field Injection**

 **Best Practice: Use Constructor Injection for immutability and testing**



Inversion of Control (IoC) Container

- **Loads bean definitions**
- **Instantiates and wires beans**
- **Manages lifecycle**



Inversion of Control (IoC) Container

Popular containers:

- **ApplicationContext** (preferred)
- **BeanFactory**

Think of it as a Smart Box that gives you the right object when needed!



Bean Configuration Styles

How to Configure Beans?

1. Annotation-based (@Component, @Autowired)
2. XML-based (Old style)
3. Java Config (@Configuration, @Bean)



Annotation-based is most modern and widely used



Recap – Spring Core in Banking Example

Putting It All Together

- **AccountService, TransactionService** – Beans
- Spring container manages object creation
- DI injects dependencies
- Easy to test and maintain

 **Spring Core = The foundation of all Spring modules!**