Understanding Core Concepts in Spring

SPRING FRAMEWORK: KEY LEARNINGS

Group ID & Artifact ID

- • Group ID is like a package name.
- • Artifact ID is similar to a class name.
- • Helps uniquely identify a Spring project and avoid conflicts in dependencies.

Why Avoid Snapshot Versions?

- • Snapshot versions are unstable as they are still in development.
- • Features may change frequently, leading to compatibility issues.
- • Use stable releases for production-ready applications.

Tight Coupling in Software Design

- • When a class is heavily dependent on another, making modifications difficult.
- • Reduces flexibility and increases maintenance effort.
- • Example:
- class GameRunner {
- MarioGame game = new MarioGame();
- game.run();
- }
- • Changing the game requires modifying GameRunner.

Example: Loose Coupling Implementation

- • Loose coupling is achieved using interfaces.
- • Example:
- interface Game { void run(); }
- class MarioGame implements Game { public void run() {…} }
- class SuperContraGame implements Game { public void run() {…} }
- class GameRunner {
- private Game game;
- GameRunner(Game game) { this.game = game; }
- void runGame() { game.run(); }
- }
- • Now, switching games is easier without modifying GameRunner.

Why Prefer Loose Coupling?

- • Enhances flexibility and makes code more maintainable.
- • Allows changing components independently.
- • Reduces dependency-related errors and improves scalability.

What is Dependency Wiring?

- • The process of providing an object with required dependencies.
- • Achieved via constructor injection or setter injection.
- • Helps in better dependency management and modular code structure.

The Role of @Configuration in Spring

- • Marks a class as a source of bean definitions.
- • Enables Spring to manage dependencies efficiently.
- • Example:
- @Configuration
- class AppConfig {
- @Bean
- public Game marioGame() { return new MarioGame(); }
- }

Managing Beans in Spring

- • context.getBean("beanName") retrieves a specific bean.
- • Multiple beans of the same type cause ambiguity.
- • Use @Bean(name = "customBean") to specify unique names.

AnnotationConfigApplicationContext

- • Initializes Spring context using Java configuration.
- • Eliminates the need for XML-based configuration.
- • Example:
- ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

Introduction to 'var' in Java

- • Introduced in Java 10.
- • Allows local variable type inference.
- • Compiler determines type at compile-time.
- • Example:
- var message = "Hello, Java!"; // Type inferred as String
- var number = 10; // Type inferred as int
- • Improves code readability but should be used judiciously.

Conclusion

- **Tight coupling** makes code rigid and hard to modify.
- **Loose coupling** improves flexibility, scalability, and maintainability.
- **Dependency Injection (DI)** allows modular and flexible code.
- **@Configuration and @Bean** help manage dependencies in Spring.
- **AnnotationConfigApplicationContext** initializes the Spring context programmatically.
- Using best practices in Spring ensures maintainable applications.
- **Using var in Java** improves code readability with type inference.