XML-based Configuration

Bean definitions and dependencies are specified in XML files. This approach provides a centralized configuration and is useful in legacy projects or when externalizing configuration is preferred. However, it can become verbose and harder to maintain in large projects.

Annotation-based Configuration

Annotations are used directly within the code to define beans and their dependencies. This approach reduces boilerplate code and is suitable for smaller projects or when a declarative style is preferred. However, it can scatter configuration information throughout the codebase.

Java-based Configuration

Java configuration uses `@Configuration` classes and `@Bean` methods to define beans. This approach provides type-safety, refactoring support, and better IDE integration. It is preferred for new projects or when programmatic configuration is desired.

When to Use Which

- **XML-based**:

  Suitable for legacy projects or when you need to change configurations without recompiling.

- **Annotation-based**:

  Ideal for smaller projects or when you prefer a declarative style and want to reduce boilerplate code.

- **Java-based**:

  Recommended for new projects, offering type safety, refactoring support, and better IDE integration.

- **Mixed**:

  You can combine these approaches within the same project, using XML for infrastructure-related beans and annotations or Java config for business logic beans.

The choice depends on project requirements, team preferences, and the need for maintainability and scalability.