Decouple from spring

Custom initialization methods, such as `@PostConstruct` or methods configured as `init-method`, are not considered coupling in Spring because they allow beans to perform necessary initialization tasks without directly depending on Spring-specific interfaces or implementations. Here's why they are not considered coupling:

1. **Decoupling from Spring interfaces:** With custom initialization methods, beans do not need to implement any specific Spring interfaces or extend Spring classes. They simply define methods annotated with `@PostConstruct` or specify initialization methods using `init-method` in XML configuration. This means that the beans are not tightly coupled to Spring's internal mechanisms.

2. **Flexible configuration:** Custom initialization methods provide flexibility in how beans are initialized. You can choose to use annotations like `@PostConstruct` for simplicity, or you can configure initialization methods using XML (`init-method`). This flexibility allows you to define initialization logic in a way that best suits your application's requirements, without being tightly bound to Spring's framework.

3. **Easy to understand:** Custom initialization methods make the initialization process explicit and easy to understand. Developers can clearly see which methods are invoked during bean initialization, enhancing code readability and maintainability. This transparency helps in understanding the bean's lifecycle without delving into Spring's internal workings.

4. **Non-invasive:** Using custom initialization methods does not require modifying existing code to adhere to Spring-specific interfaces or patterns. You can simply add initialization logic to your existing beans by annotating methods or configuring them in XML, without introducing dependencies on Spring-specific classes or interfaces.

Overall, custom initialization methods in Spring provide a way to perform bean initialization tasks in a non-invasive, flexible, and easy-to-understand manner, without tightly coupling beans to Spring's internals. This approach aligns with the principles of loose coupling and separation of concerns, making the application more maintainable and adaptable.