Injeção de dependência com Spring

A injeção de dependência é um padrão de projeto que visa desacoplar classes e aumentar a flexibilidade do código.

O spring utiliza um container IoC para gerenciar as dependências entre as classes. O uso de um container IoC torna o código mais flexível e modular, pois, as classes não dependem diretamente de implementações concretas. Isso facilita a troca de implementações e a criação de testes unitários.

Benefícios do uso de um container IoC:

1. Acoplamento baixo: As classes dependem de interfaces, não de implementações concretas.
2. Reusabilidade: As dependências podem ser facilmente reutilizadas em diferentes partes da aplicação.
3. Manutenabilidade: O código torna-se mais modular e fácil de entender, com responsabilidades bem definidas.
4. Testabilidade: Os testes unitários se tornam mais fáceis de escrever e executar, pois as dependências podem ser facilmente mockadas.

Ex: **Sem IoC**

Imagine que uma aplicação que possui uma classe "ClienteService" que precisa de um repositório para acessar dados do cliente. O repositório pode ser implementado por diferentes classes, como "ClienteRepositoryJdbc" ou "ClienteRepositoryJpa".

Injeção de Dependência com Spring

```
Ex: 1

public class ClienteService {

    private ClienteRepository repository;

    public ClienteService() {
        this.repository = new ClienteRepositoryJdbc(); // Injeção manual
    }

    public List<Cliente> buscarTodos() {
        return repository.findAll();
    }
}
```

Neste caso, a classe "ClienteService depende diretamente da implementação "Cliente RepositoryJdbc". Isso torna o código menos flexível, pois a classe "Cliente Service" precisa ser modificada caso seja necessário usar uma implementação diferente do repositório.

Ex: **Com IoC**

Neste caso, a classe "Cliente Service" não dependerá diretamente de nenhuma implementação do repositório, apenas da interface.

Injeção de Dependência com Spring

```
23
24      public class ClienteService {
25
26          @Autowired
27          private ClienteRepository repository;
28
29          public List<Cliente> buscarTodos() {
30              return repository.findAll();
31          }
32      }
33
34      @Repository
35      public interface ClienteRepository {
36
37          List<Cliente> findAll();
38      }
39
40      @Component
41      public class ClienteRepositoryJdbc implements ClienteRepository {
42
43          // Implementação do método findAll() usando JDBC
44      }
45
46      @Component
47      public class ClienteRepositoryJpa implements ClienteRepository {
48
49          // Implementação do método findAll() usando JPA
50      }
```

No exemplo acima, o container IoC cria instâncias das classes "ClienteService", "ClienteRepositoryJdbc" e "ClienteRepositoryJpa". Ele também injeta a instância de "ClienteRepositoryJdbc" na classe "ClienteService".

Mais informações sobre Spring...

O Spring utiliza um container IoC para gerenciar as dependências entre as classes. O container IoC é responsável por:

- Criar instâncias das classes que são gerenciadas pelo container (os "beans").
- Injetar as dependências nas classes que os necessitam.
- Gerenciar o ciclo de vida dos beans.