03/09/2024

@*Configuration*

@*Profile*("test")

public class TestConfig implements *CommandLineRunner* {

    /\*

    a interface CommandLineRunner vai simular uma main

    é necessário implementar o método run, que

     \*/

    @*Autowired*

    private *UserRepository* userRepository;

    @*Override*

    public *void* run(*String*... *args*) throws *Exception* {

*User* u1 = new User(null, "Maria Brown", "maria@gmail.com", "988888888", "123456");

*User* u2 = new User(null, "Alex Green", "alex@gmail.com", "977777777", "123456");

        userRepository.saveAll(Arrays.asList(u1, u2));  // persiste no banco de dados

    }

}

Sobre database seeding, resposta com ChatGPT:  
  
No contexto apresentado, o código está utilizando a técnica de \*\*database seeding\*\* para preencher o banco de dados com dados iniciais específicos para um perfil de ambiente de teste. Aqui está uma explicação detalhada sobre o que o código faz e como se relaciona com o conceito de database seeding:

### Contexto do Código

1. \*\*Anotação `@Configuration`:\*\*

- Indica que a classe `TestConfig` é uma classe de configuração do Spring, ou seja, ela pode definir beans e configurações que serão usadas pela aplicação.

2. \*\*Anotação `@Profile("test"):\*\*

- Especifica que esta configuração será ativada somente quando o perfil "test" estiver ativo. No Spring, perfis permitem que diferentes configurações sejam aplicadas dependendo do ambiente (desenvolvimento, teste, produção, etc.).

3. \*\*Interface `CommandLineRunner`:\*\*

- O `CommandLineRunner` é uma interface funcional do Spring Boot que é usada para executar um pedaço de código logo após a inicialização da aplicação. O método `run` é chamado automaticamente com os argumentos da linha de comando (se houver), permitindo a execução de lógica personalizada.

- Nesse caso, o `CommandLineRunner` está sendo usado para executar o código de seeding assim que a aplicação inicia no ambiente de teste.

4. \*\*Injeção de Dependência com `@Autowired`:\*\*

- O repositório `UserRepository` está sendo injetado na classe `TestConfig`. Esse repositório é provavelmente uma interface que estende o `JpaRepository` ou `CrudRepository`, fornecendo métodos de acesso ao banco de dados.

5. \*\*Método `run(String... args)`:\*\*

- Esse método é onde o seeding do banco de dados ocorre. No exemplo fornecido, dois objetos `User` são criados (`u1` e `u2`) com informações como nome, email, telefone e senha.

- Esses usuários são então persistidos no banco de dados através do método `saveAll` do `userRepository`.

### Como isso se relaciona com Database Seeding

- \*\*Database Seeding:\*\* No contexto do código, o database seeding está sendo utilizado para criar e persistir dois usuários (`Maria Brown` e `Alex Green`) no banco de dados assim que a aplicação é iniciada, mas apenas quando o perfil de "test" estiver ativo. Isso é útil em ambientes de teste porque fornece um conjunto consistente de dados para validação e testes automatizados.

- \*\*Benefícios do Database Seeding:\*\*

- \*\*Consistência:\*\* Garante que, em cada execução dos testes, o banco de dados comece com um estado conhecido, o que facilita a depuração e a reprodutibilidade dos testes.

- \*\*Automatização:\*\* Automatiza o processo de configuração do ambiente de teste, reduzindo a necessidade de configuração manual e possibilitando a integração contínua.

- \*\*Facilidade de Teste:\*\* Permite que os desenvolvedores escrevam testes que dependem de dados específicos no banco de dados sem precisar inserir esses dados manualmente antes de cada teste.

### Considerações Finais

O uso de seeding em ambientes de teste é uma prática comum em desenvolvimento de software, especialmente em aplicações que utilizam frameworks como Spring Boot. Ele permite que os desenvolvedores mantenham um ambiente de teste controlado e previsível, essencial para a qualidade do software.