Desafio 10

Respostas

1. O **Spring Boot** é um framework Java open source que tem como objetivo facilitar os processos de configurações e otimizar o tempo em aplicações Java.
2. Spring Web, Spring Data JPA e Spring Actuator. Quanto ao Spring Web pode-se dizer que essa dependência facilita a comunicação da aplicação com a web através de requisições HTTP, já o Spring Data cuida do relacionamento com o banco de banco e o Spring Actuator nos permite saber a saúde e métrica da aplicação.
3. Até o momento foram utilizados o Spring Web, Spring Data JPA, o Spring Dev Tools , H2, Lombok e MapStruct.
4. Vantagens:
   1. O maior benefício de usar o Spring Boot é conseguir otimizar seu tempo e aumentar sua produtividade, já que ele simplifica o desenvolvimento de aplicações;
   2. Possui um servidor embarcado(o TomCat);
   3. Ajuda na organização do pom.xml;
   4. Facilita a injeção de dependências;
   5. Auxilia com tarefas do dia a dia , iniciando e atualizando a aplicação através do Spring Dev tools.
5. Primeiramente, acesse o site do [spring initializr](https://start.spring.io/" \t "_blank) e preencha as informações:

* **Project:** Selecione a opção Maven ou Gradle
* **Language:** Selecione a opção Java
* **Spring Boot:** A versão mais recente
* **Group:** Configure de acordo sua preferência
* **Artifcat:** Configure de acordo sua preferência
* **Packaging:** Jar
* **Java:** 11 ou outra versão

Em seguida, selecione as dependências necessárias para o desenvolvimento do projeto e faça o download. Para fazer uma conexão com banco SQL Server é preciso configurar o application.properties da seguinte maneira:

|  |
| --- |
| spring.datasource.url=jdbc:sqlserver://localhost;databaseName=springbootdb |
| spring.datasource.username=as |
| [spring.datasource.password=](mailto:spring.datasource.password=Projects@123) 123456 |
| spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver |
| spring.jpa.show-sql=true  spring.jpa.hibernate.ddl-auto = create-drop |
| spring.jpa.hibernate.dialect=org.hibernate.dialect.SQLServer2012Dialect |
|  |
| <!-- https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc -->  <dependency>  <groupId>com.microsoft.sqlserver</groupId>  <artifactId>mssql-jdbc</artifactId>  <version>9.5.0.jre8-preview</version>  </dependency> |
|  |
|  |

f) O pom.xml serve para efetuarmos a configuração da aplicação, bem como a importação de dependências e subdependências.

g) No application.properties , estabelecemos a comunicação com o banco de dados que por sua vez, pode ser relacional ou não-relacional .

h) O uso de anotações varia de acordo com a funcionalidade dela, no geral, elas podem auxiliar na injeção de dependência, como por exemplo o uso do @Autowired.

i)

* 1. @Controller

Essa annotation é usada para criar uma classe como um controlador (**controller**), que pode manipular solicitações (**requests**) do cliente e enviar uma resposta (**response**) de volta ao cliente.

# 2. @RequestMapping

É uma annotation no nível do método que é colocada sobre um método. Ele fornece o mapeamento entre o caminho da solicitação e o método manipulador. Ele também suporta algumas opções avançadas que podem ser usadas para especificar métodos separados para diferentes tipos de solicitação no mesmo URI.

# 3. @RequestParam

Essa é outra anotação muito útil do Spring MVC, usada para mapear os parâmetros HTTP a argumentos de métodos.

# 4. @PathVariable

Essa é outra anotação usada para recuperar dados da URL.

# 5. @RequestBody

Essa annotation pode converter dados de entrada em objetos Java passados para o método do controlador.

# 6. @ResponseBody

Essa annotaion é usada para transformar um objeto Java retornado do **controller**em uma representação de recurso solicitada por um cliente **REST**.

# 7. @RestController

# Essa é uma annotation muito conveniente para o desenvolvimento de serviços **RESTful**com o **Framework Spring MVC**.

# 8. @SpringBootApplication

# Essa annotation única combina três anotações como **@Configuration**, **@EnableAutoConfiguration** e **@ComponentScan**. Através do uso do **Spring Boot**, podemos executar a aplicação sem a necessidade de instalarmos em um servidor Web, pois ele vem com um servidor Tomcat incorporado.

# 9. @EnableAutoConfiguration

# Essa é outra annotation de inicialização do **Spring**que ativa o recurso de configuração automática, o que faz com que o **Spring **“****adivinhe” a configuração baseada nos JAR apresentados no classpath.

# 10. @ResponseStatus

# Esta annotation pode ser usada para substituir o código de resposta HTTP de uma resposta(response).

# j) O Advice em Spring é uma ação realizada por um aspecto em um ponto de junção específico. Há diferentes tipos de advices como “ao redor”, “antes” e “depois”. O principal objetivo do uso deles é oferecer suporte a questões transversais, como registro, criação de perfil, armazenamento em cache e gerenciamento de transações.

# Tipos de Advice:

# Before Advice

# Como o nome indica, é executado antes do ponto de junção. Isso não impede a execução contínua do método recomendado, a menos que uma exceção seja lançada.

# After Advice

# É declarado usando a anotação @After, é executado após a execução de um método correspondente, independentemente de uma exceção ter sido lançada ou não.

# Around Advice

# Este é o tipo de advice mais poderoso. O advice Around pode realizar um comportamento personalizado antes e depois da invocação do método. Ele também é responsável por escolher se deve prosseguir para o ponto de junção ou atalho para a execução do método recomendado fornecendo seu próprio valor de retorno ou lançando uma exceção.

# k) IoC também é conhecido como injeção de dependência (DI). É um processo pelo qual os objetos definem suas dependências.

# l) Através do uso Spring Security temos acesso a vários recursos de autenticação e de autorização que protegem a nossa aplicação.

# m) O pacote responsável pela conexão com o banco de dados é o Spring Data.

# n)Conforme , vemos no exemplo abaixo uma task serve mostrar determinadas atividades que serão realizadas através da nossa aplicação a depender do período de tempo configurado. Nesse sentido, fiz uma task à qual deve ser mostrada a cada 2 segundos. Assim, criei uma nova classe, marquei com a anotação @Component para indicar que ela usará o padrão de injeção de dependência e seguida marquei o método startTask1 com a anotação @Scheduled para configurar o tempo de execução da task que será a cada 2 segundos, como já citado anteriormente.

# Interface gráfica do usuário, Texto, Aplicativo Descrição gerada automaticamente