Índice

Assuntos: Injeção de dependências com Spring Framework, criação de plano de carreira.

Abertura Alunos:
Apresentação pitch
Anderson e Leo Marques.

Spring Framework

JPA

Java Persistence API (JPA) é uma especificação do Java que facilita o mapeamento objeto-relacional e o acesso a bancos de dados relacionais em aplicações Java, oferecendo uma abordagem mais simples para interagir com bases de dados por meio de objetos Java Ele é a interface que manipula os dados da sua API e não cria métodos, ele usa métodos já criados. Ele usa os dois tipos de banco de dados relacional e não relacional.

JPA é uma biblioteca que recupera e usa que estão salvos no Banco de Dados. Ele é responsável por fazer todas as instruções SQL sem que precisemos escrever uma única linha em nosso código SQL.

Mapeamento Objeto Relacional

Mapeamento Objeto Relacional é a representação de uma Tabela de um Banco de dados Relacional (MySQL, PostgreSQL, Oracle, SQL Server e etc), através de Classes Java, que dentro do contexto Spring e seguindo o Modelo MVC, são implementadas na Camada Model. Essa técnica de mapeamento também é conhecida como ORM ou Object Relational Mapping:

Banco de Dados	Linguagem Orientada a Objetos
Tabela	Classe
Coluna	Atributo
Registro	Objeto

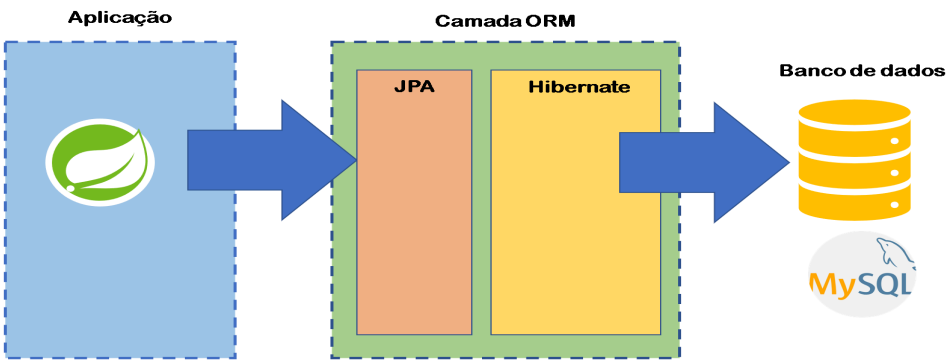
Registro ou linha

Com as annotations o Java é capaz de registrar as características de uma tabela.

Hibernate

Também é um Framework que faz um mapeamento do nosso objeto, ele tira a complexidade de JAVA para traduzir para o banco de dados.

Hibernate é um framework de mapeamento objeto-relacional em Java, facilitando a interação entre objetos Java e bancos de dados relacionais. Ele oferece uma solução de persistência eficiente e simplificada, permitindo que desenvolvedores realizem operações de banco de dados usando objetos Java.



Comandos blog pessoal

spring.jpa.hibernate.ddl-auto=update

Comando para definir como o JPA inicializará o banco de dados

Ele compara os dados com o banco de dados existente e se houver diferença, ele atualiza.

spring.jpa.database=mysql

Comando que define qual o database a ser utilizado

spring.datasource.url=jdbc:mysql://localhost/db_blogpessoal?createDatabaseIfNotExist=true&serverTimezone=America/Sao_Paulo&useSSl=false

Definição dos dados da conexão com o Banco de dados:

- **jdbc:mysql://localhost/db_blogpessoal** ⇨ endereço (jdbc:mysql://localhost/) + nome do Banco (db_blogpessoal)
- **?createDatabaseIfNotExist=true** ⇨ criar automaticamente o Banco de dados no MySQL caso ele não exista (true)

  createDatabaseIfNotExist=true ⇨ crie esse banco caso não exista
- **&serverTimezone=America/Sao_Paulo** ⇨ define o fuso horário do servidor MySQL (America/Sao_Paulo)
- **&useSSl=false** ⇨ desabilita a camada de segurança da conexão com o MySQL (SSL)

spring.datasource.username=USERNAME

Define o usuário no MySQL

spring.datasource.password=PASSWORD

Define a senha a ser usada no MySQL

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

Informa o nome do Driver do Banco de dados.

spring.jpa.show-sql=true

Todas as Queries enquanto executadas vão ser exibidas.

P.S: Durante o desenvolvimento da aplicação, você pode manter a opção: spring.jpa.show-sql habilitada (true). Em produção ela deve ser desabilitada.

spring.jpa.properties.hibernate.dialect =org.hibernate.dialect.MySQLDialect

Define que usaremos o dialeto do hibernate para SQL, convertendo de HQL para SQL , pois o hibernate não é ligado a nenhum banco de dados específico.

spring.jpa.properties.jakarta.persistence.sharedCache.mode=ENABLE_SELECTIVE

A Biblioteca Jakarta é uma atualização das bibliotecas JAVAX no JPA. Ele auxilia na compatibilidade para gerar a configuração do hibernate.

spring.jackson.date-format=yyyy-MM-dd HH:mm:ss

Configura o formato da Data (AAAA-MM-dd) e da Hora (HH:mm:ss) da aplicação.

spring.jackson.time-zone=Brazil/East

Configura a Time Zone

ATENÇÃO: Não insira comentários no arquivo application.properties. Ao final do Bloco 2 faremos o Deploy na nuvem e geralmente comentários causam erros de compilação do projeto durante o Deploy

Anexos de Erros

Erro	Descrição
Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured.	A configuração da conexão com o Banco de dados não foi implementada. Configure a conexão com o Banco de dados no

	arquivo application.properties. Caso o erro persista, experimente apagar a pasta .m2, localizada em: C:\Users\<seu_usuario>\.m2
Access denied for user 'root'@'localhost' (using password: YES)	O nome do usuário ou a senha do seu Banco de dados está incorreta. Verifique qual foi a senha que você cadastrou na instalação do seu MySQL e altere no arquivo application.properties

Camadas BLOG PESSOAL

Camada	Descrição
Model	Camada responsável pela abstração dos nossos Objetos em registros das nossas tabelas, que serão geradas no Banco de dados. As Classes criadas nesta camada representam os objetos que serão persistidos no Banco de dados.
Repository	Camada responsável por implementar as Interfaces, que contém diversos Métodos pré-implementados para a manipulação de dados de uma entidade, como Métodos para salvar, deletar, listar e recuperar dados da Classe. Para criar estas Interfaces basta Herdar (extends) a Interface JpaRepository.
Controller	Camada responsável por receber todas as Requisições HTTP (HTTP Request), enviadas por um Cliente HTTP (Insomnia, Postman ou o Front-end da aplicação), para a nossa aplicação e responder (HTTP Response) as requisições de acordo com o resultado do processamento da requisição no Back-end.

Comandos Model - Postagem

Não deixe espaço(pular linha) entre as anotações e as variáveis, atributos.

@Entity

Annotation que vai gerar uma tabela no meu banco de dados da minha aplicação
A biblioteca correta é a Jakarta.

@Table (name= “tb_postagens”)

Annotation que nomeia a tabela que você cria com @Entity. A biblioteca correta é Jakarta.
Se não usar a annotation table ele cria a tabela com o mesmo nome da classe.

LocalDateTime

Tipo de variável que usa data e horário locais.

@Id

Annotation que indica que um campo é uma chave primária da tabela indicada

@GeneratedValue (strategy = GenerationType.IDENTITY)

Annotation que indica que um campo é auto incrementado.

GeneratedValue: Indica que o valor da chave primária será gerado automaticamente.

Strategy: Atributo que faz parte da anotação @GeneratedValue. Nesse contexto, "strategy" especifica a estratégia a ser usada para gerar valores para a chave primária.

strategy = GenerationType.IDENTITY: Define a estratégia de geração como "IDENTITY", que geralmente significa que o banco de dados cuidará automaticamente da geração dos valores das chaves primárias, usando um recurso chamado "IDENTITY" ou equivalente, que fornece incremento automático.

@NotBlank(message = “mensagem”)

Indica que o atributo é de preenchimento obrigatório.

Obs: (message) não é obrigatório.

@Size (min = x, max = x, message = “mensagem”)

Annotation que indica o tamanho de um atributo de uma tabela.

Obs: (message) não é obrigatório.

@UpdateTimestamp

Annotation que configura a data e hora da coluna com a data e hora do sistema operacional.

Anexo II - Estratégias para geração da Chave Primária

Strategy	Descrição
GenerationType.AUTO	Valor padrão, deixa com o provedor de persistência a escolha da estratégia mais adequada de acordo com o Banco de dados.
GenerationType.IDENTITY	Informamos ao provedor de persistência que os valores a serem atribuídos ao identificador único serão gerados pela coluna de auto incremento do banco de dados. Assim, um valor para o identificador é gerado para cada registro inserido no banco. Alguns bancos de dados podem não suportar essa opção.
GenerationType.SEQUENCE	Informamos ao provedor de persistência que os valores serão gerados a partir de uma sequence. Caso não seja especificado um nome para a sequence, será utilizada uma sequence padrão, a qual será global, para todas as entidades. Caso uma sequence seja especificada, o provedor passará a adotar essa sequence para criação das chaves primárias. Alguns bancos de dados podem não suportar essa opção, como o MySQL por exemplo.
GenerationType.TABLE	Com a opção TABLE é necessário criar uma tabela para gerenciar as chaves primárias. Por causa da sobrecarga de consultas necessárias para manter a tabela atualizada, essa opção é pouco

	recomendada

Persistência - Criar no Banco de Dados.

Comandos Model - Postagem Controller

@RestController

Ela define que é uma classe controladora

@RequestMapping

Essa anotação é usada para mapear solicitações HTTP para métodos de manipulação dentro de um controlador.

@CrosssOrigin (origins = “*”, allowedHeaders = “*”)

Annotation que indica que a classe controladora permite que possa receber requisições vindas de várias fontes, ela é essencial para dar acesso a todos os métodos e recursos dessa aplicação.

- **origins = “*”** - de qualquer origem
- **allowedHeaders = “*”** qualquer cabeçalho de requisição é aceito

@Autowired

Annotation que é como uma "mãozinha mágica" do Spring: quando você coloca essa anotação em um campo, método ou construtor de uma classe, o Spring automaticamente encontra o objeto apropriado (por exemplo, uma dependência) e o "injeta" no seu código, sem que você precise se preocupar em criá-lo manualmente. É como ter um assistente que fornece automaticamente as ferramentas que você precisa para fazer o seu trabalho.

JpaRepository

É uma biblioteca que tem métodos pré definidos.

Comandos JpaRepository

Método	Descrição
save(Objeto objeto)	Cria ou Atualiza um objeto no Banco de Dados.
findById(Long id)	Retorna (exibe) um Objeto persistido de acordo com o id informado.

existsById(Long id)	Retorna True se um Objeto identificado pelo id estiver persistido no Banco de dados.
findAll()	Retorna (exibe) todos os Objetos persistidos.
deleteById(Long id)	Localiza um Objeto persistido pelo id e deleta caso ele seja encontrado. Não é possível desfazer esta operação.
deleteAll()	Deleta todos os Objetos persistidos.Não é possível desfazer esta operação.

Links e afins

CookBook :

https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/04_spring/03.md

https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/04_spring/04.md

https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/04_spring/05.md

Criação do DER:

https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/03_mysql/04.md

Annotations:

https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/04_spring/guia_jpa.md

---

Dúvidas