WEBSERVICES REST ASAP COM SPRING DATA JPA

DANIEL ABELLA

UNIFACISA

CONTEÚDO

INTRODUÇÃO

spring Data JPA é parte do framework Spring Data, na qual facilita a criação de webservices baseados em Repositories JPA.

Este framework aprimora as camadas de acesso à dados, reduzindo o esforço da quantidade de implementação necessária atualmente.

FUNCIONALIDADES

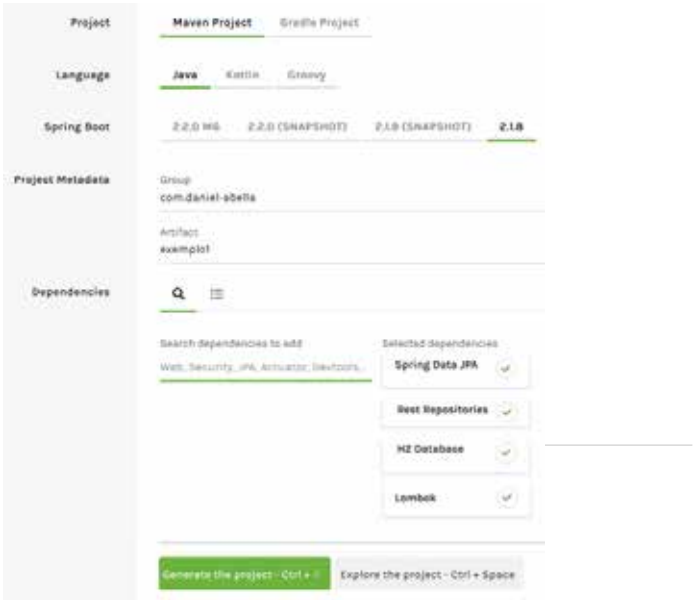
- Suporte sofisticado para construir repositories baseado em Spring e JPA
- Suporte para predicados QueryDSL e type-safe JPA Queries
- Auditoria Transparante na Entidade
- Suporte à paginação e execução dinâmica de consultas
- Validação de consultas anotadas pelo @Query no momento da inicialização
- Suporte para mapeamento de entidade baseado em XML
- Configuração do Repository baseado em JavaConfig, introduzindo @EnableJpaRepositories

INICIANDO PROJETO COM SPRING DATA JPA

Para criar o projeto, devemos acessar o Spring Initializer por meio do endereço https://start.spring.io/.

Devemos adequadamente preencher os campos Group e Artifact na seção Project Metadata. Por fim, na seção Dependencies, devemos preencher as seguintes dependencias: Spring Data JPA, Rest Repositories, H2 Database e Lombok. O resultado deverá semelhante ao da imagem a seguir.

Ao clicar no botão “Generate Project”, um projeto Maven compactado em um .zip é disponibilizado. Deve descompactar este arquivo e importar como projeto Maven na IDE de sua preferência.



IMPLEMENTAÇÃO

DE UM PROJETO COM SPRING DATA JPA

INTRODUÇÃO

Neste exemplo, criaremos um WebService REST para uma Person com serviços GET/{id}, GET, POST, PUT/{id} e DELETE/{id}. O projeto consistirá em apenas dois arquivos: Person.java, a entidade JPA e o PersonRepository.java, que representa o nosso Rest Repository.

ENTIDADE

Entidade Person com os campos ID (Autogerado), Firstname e Lastname. Usamos o framework Lombok para gerar os getters e setters usando as anotações @Getter e @Setter e gerar construtor vazio usando @NoArgsConstructor.

```
import javax.persistence.*;
import lombok.*;

@Entity @NoArgsConstructor @Getter @Setter
public class Person {

    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    private String firstName;
    private String lastName;
}
```

REST REPOSITORY

Que inicie a mágica! Criaremos o nosso Repository e como um passe de mágica os endpoints básicos serão gerados. No código abaixo, verificamos a anotação @RepositoryRestResource.

```
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource(collectionResourceRel = "people", path = "people")
public interface PersonRepository
    extends PagingAndSortingRepository<Person, Long> {

}
```

- As propriedades collectionResourceRel e path devem ser preenchidas adequadamente com os valores para que os endpoints sejam acessíveis.
- A interface deve estender PagingAndSortingRepository
- <Person, Long> se referem respectivamente a entidade que o Repository em questão e o tipo do campo anotado com @Id da entidade.

EXECUÇÃO

Execute a classe Exemplo1Application.java e pronto, nosso WebService está funcionando na porta 8080.

CÓDIGO FONTE (REPOSITÓRIO GIT)

O código fonte do projeto está no endereço: https://github.com/daniel-abella/unifacisa-lsi/exemplo1

REFERÊNCIAS

https://www.baeldung.com/spring-data-rest-intro
https://www.devmedia.com.br/spring-boot-como-criar-um-servidor-rest-com-spring-data/34006
https://www.devmedia.com.br/spring-boot-como-criar-um-servidor-rest-com-spring-data/34006
https://docs.spring.io/spring-data/rest/docs/current/reference/html/