



# EducaCiência FastCode

Fala Galera,

- Artigo: 33/2020 Data: Outubro/2020
- Público Alvo: Desenvolvedores – Iniciantes
- Tecnologia: Java
- Tema: Spring Boot Método Update + documentação Swagger
- Link: <https://github.com/perucello/DevFP>

Neste artigo, daremos abordaremos Spring Boot e iremos mapear o CRUD com repositório CRUD Repository.

Este artigo é uma continuação do artigo 32/2020 onde teremos um total 4 artigos para concluir nosso propósito , sendo este 33/2020 o primeiro.

Já temos disponível:

- ⇒ 31/2020 – Select + Swagger
- ⇒ 32/2020 – inserir + Swagger

Para este ambiente , já temos criado nosso Banco de Dados MySql e as persistências.

Nosso ambiente consiste em:

- ⇒ Banco de Dados MySql

```
1  
2 • create database EducaSpring;  
3 • use EducaSpring;  
4
```

Output

Action Output

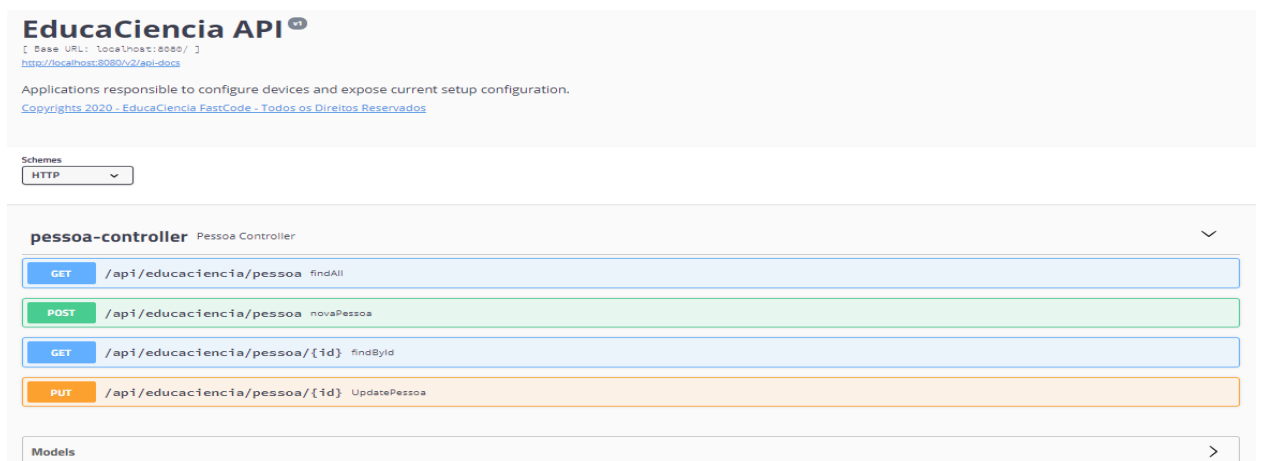
#	Time	Action
1	13:20:44	create database EducaSpring
2	13:20:49	use EducaSpring



Para criarmos nosso método Atualizar e seu endpoint , vamos dar continuidade em nosso projeto apenas manipulando a classe PessoaController.

```
1 package com.educaspring.EducaSpring.controllers;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping(path = "/api/educaciencia/pessoa")
7 public class PessoaController {
8
9     private static final Pessoa Ok = null;
10
11     @Autowired
12     private PessoaRepository pessoaRepository;
13
14     @PostMapping
15     public @ResponseBody Pessoa novaPessoa(@Validated Pessoa pessoa) {
16         pessoaRepository.save(pessoa);
17         return pessoa;
18     }
19
20     @GetMapping
21     public @ResponseBody ResponseEntity<List<Pessoa>> findAll() {
22         List<Pessoa> list = (List<Pessoa>) pessoaRepository.findAll();
23         return ResponseEntity.ok().body(list);
24     }
25
26     @GetMapping(value =("/{id}")
27     public @ResponseBody ResponseEntity<Optional<Pessoa>> findById(@PathVariable Integer id) {
28         Optional<Pessoa> obj = pessoaRepository.findById(id);
29         return ResponseEntity.ok().body(obj);
30     }
31
32     @PutMapping(value =("/{id}")
33     @Transactional
34     public @ResponseBody Pessoa UpdatePessoa(@PathVariable Integer id, @RequestBody Pessoa pessoa) {
35         pessoaRepository.save(pessoa);
36         return pessoa;
37     }
38 }
```

Feito este procedimento, podemos salvar e iniciar nosso sistema para testarmos. Ao carregar o Spring Boot, e iniciarmos nossa documentação Swagger, já nos apresenta o método PUT que é responsável pela atualização de dados.





Como temos criado esta documentação, vamos manipulá-la !  
Na opção Put do Swagger , vamos inserir os dados para atualizá-lo e executá-lo.

**PUT** /api/educaciencia/pessoa/{id} UpdatePessoa

Parameters

Name	Description
<b>id</b> * required integer (\$int32) (path)	id <input type="text" value="1"/>
<b>pessoa</b> * required (body)	pessoa <div><pre>{   "email": "testeUpdate",   "id": 1,   "nome": "testeUpdate" }</pre></div>

Parameter content type  
application/json

Execute

Curl

```
curl -X PUT "http://localhost:8080/api/educaciencia/pessoa/1" -H "accept: */*" -H "Content-Type: application/json" -d '{"email": "testeUpdate", "id": 1, "nome": "testeUpdate"}'
```

Request URL  
http://localhost:8080/api/educaciencia/pessoa/1

Server response

Code	Details
200	<div><p>Response body</p><pre>{   "id": 1,   "nome": "testeUpdate",   "email": "testeUpdate" }</pre></div> <div><p>Response headers</p><pre>cache-control: no-cache, no-store, max-age=0, must-revalidate connection: keep-alive content-type: application/json date: Tue, 27 Oct 2020 22:46:41 GMT expires: 0 keep-alive: timeout=60 pragma: no-cache transfer-encoding: chunked x-content-type-options: nosniff x-frame-options: DENY x-ssr-protection: 1; mode=block</pre></div>

Responses

Code	Description
200	OK





Vamos na Opção GET e certificar-nos da atualização !

**pessoa-controller** Pessoa Controller

GET /api/educaciencia/pessoa findAll

Parameters

No parameters

Execute Clear

Responses

Response content type \*/\*

Curl

```
curl -X GET "http://localhost:8080/api/educaciencia/pessoa" -H "accept: */*"
```

Request URL

```
http://localhost:8080/api/educaciencia/pessoa
```

Server response

Code	Details
200	<p>Response body</p> <pre>{   "id": 1,   "nome": "testelupdate",   "email": "testelupdate" }</pre>

Download

swagger Select a spec default

## EducaCiencia API

[ Base URL: localhost:8080/ ]  
<http://localhost:8080/v2/api-docs>

Applications responsible to configure devices and expose current setup configuration.  
[Copyrights 2020 - EducaCiencia FastCode - Todos os Direitos Reservados](#)

Schemes

HTTP

**pessoa-controller** Pessoa Controller

GET /api/educaciencia/pessoa findAll

POST /api/educaciencia/pessoa novaPessoa

GET /api/educaciencia/pessoa/{id} findById

PUT /api/educaciencia/pessoa/{id} UpdatePessoa

Models

Optional«Pessoa» {  
 present boolean  
}

Pessoa {  
 email string  
 id integer(\$int32)  
 nome string  
}

Vimos no entanto que nosso código funcionou como esperado, saliento que o artigo 31/2020 é o início da nossa sequência de Spring Boot, fiquem ligados nos sequenciais onde os dados são





baseados no curso de Java que ministro na Escola Evolua – Ensino Profissionalizante e como proposito de ajuda à comunidade, estamos trazendo parte da didática em forma de artigo comunitário e assim podemos contribuir com a comunidade Tecnológica como um todo.

Agradeço imensamente a Diretoria da Escola Evolua de Sumaré.

Os códigos estarão disponíveis no Git.

Até mais !

Espero ter ajudado !

