



Desenvolvimento de Serviços com Spring Boot

Etapa/Aula 03.1

Professor(a): Flávio Neves

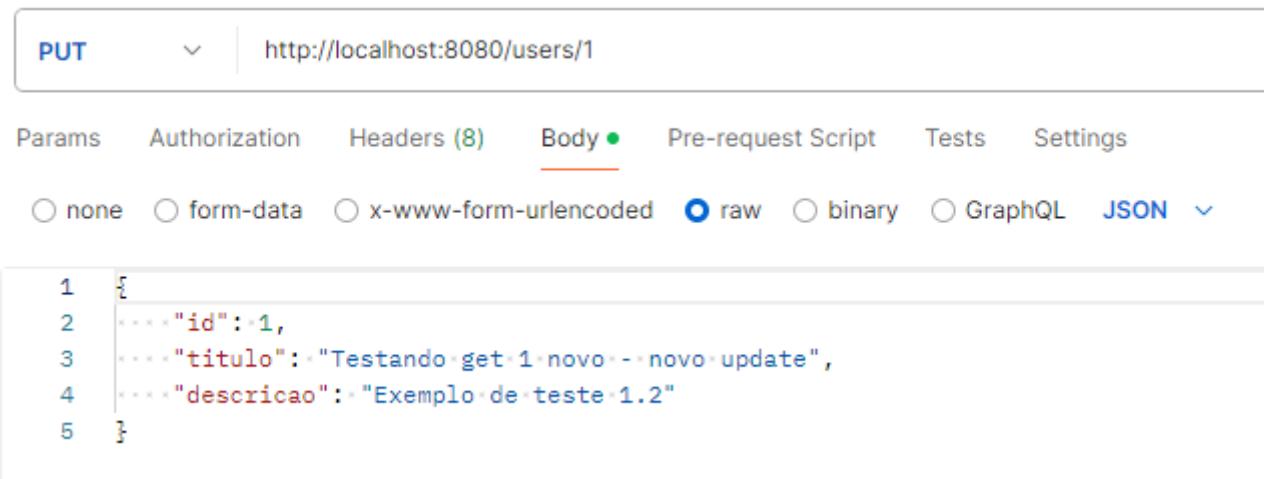
E-mail: flavio.neves@prof.infnet.edu.br

Roteiro da Aula

- Funcionamento das Requisições PUT
- Funcionamento das Requisições DELETE
- Exemplo de uso dos métodos PUT e DELETE
- Consumo de uma API

APIs REST- Requisições PUT

- As requisições PUT são utilizadas para atualizar todo o recurso existente.



PUT <http://localhost:8080/users/1>

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1 {
2     "id": 1,
3     "titulo": "Testando get 1 novo -- novo update",
4     "descricao": "Exemplo de teste 1.2"
5 }
```

APIs REST- Requisições PUT

- Regras para requisições PUT:
 - Quando o recurso for encontrado e atualizado o código da resposta HTTP deve ser 200(OK), o recurso atualizado deve ser retornado no corpo da requisição, e deve conter os links de acessos para consulta desse recurso.
 - Caso o recurso não seja encontrado, o código da resposta HTTP deve ser 404 (NOT FOUND).
 - Caso o recurso recebido no corpo da requisição não esteja de acordo com o esperado pelo servidor, o código da resposta HTTP deve ser 400 (BAD REQUEST);

APIs REST- Requisições DELETE

- As requisições DELETE são utilizadas para excluir um recurso.

DELETE http://localhost:8080/users/1

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1 {
2     "id": 1,
3     "titulo": "Testando get 1 novo - novo update",
4     "descricao": "Exemplo de teste 1.2"
5 }
```

APIs REST- Requisições DELETE

- Regras para requisições DELETE:
 - Se o recurso existir e a operação for bem sucedida e nenhum recurso precisa ser retornado, o código HTTP retornado deve ser 204 (NO CONTENT), mas caso um recurso seja retornado o código HTTP deve ser 200(OK), e o recurso deve ser adicionado no corpo da resposta.
 - Caso o recurso não seja encontrado, o código da resposta HTTP deve ser 404 (NOT FOUND).



Getters e Setters

- Em Java, os métodos getters e setters são usados para acessar e modificar os atributos (variáveis de instância) de uma classe de maneira controlada.
- Eles são uma parte importante do conceito de encapsulamento, que ajuda a proteger os dados de uma classe e a controlar o acesso a esses dados.

@PathVariable

- A anotação **@PathVariable** é utilizada para obter dados do caminho do URL.
- Ao definir marcadores de posição no URL de mapeamento do pedido, pode associar esses marcadores de posição a parâmetros de método anotados com **@PathVariable**.
- Isto permite-lhe aceder a valores dinâmicos a partir do URL e utilizá-los no seu código.
- Por exemplo, pode extrair um ID de utilizador de um URL como /users/123 e passá-lo para um método que recupera os detalhes do utilizador correspondente.
- Traduzido com a versão gratuita do tradutor - DeepL.com

- **@Autowired** é a anotação mais utilizada com relação a injeção de dependências.
- **@RestController**: marca a classe como um controlador em que cada método devolve um objeto de domínio
- **@RequestMapping**: anotação para mapear pedidos para métodos de controladores

- **@GetMapping**: Anotação para o mapeamento de pedidos HTTP GET para métodos de tratamento específicos.
- **@PutMapping**: O PUT é semelhante ao POST na medida em que pode criar recursos, mas quando existe um URI definido.
- **@ResponseBody**: Método para que o retorno seja serializado para o corpo da resposta através de um `HttpMessageWriter`.

RestTemplate

- O RestTemplate é normalmente utilizado como um componente partilhado.
- No entanto, a sua configuração não suporta a modificação simultânea e,
- como tal, a sua configuração é normalmente preparada no arranque. Se necessário,
- é possível criar várias instâncias de RestTemplate configuradas de forma diferente no arranque.
- Essas instâncias podem utilizar o mesmo ClientHttpRequestFactory subjacente se precisarem de partilhar recursos de cliente HTTP.
- Traduzido com a versão gratuita do tradutor - DeepL.com

- `getForEntity`: Método para consumir a API