

## Introdução ao Uso de Serviços Web Usando REST

### - Parte 2 -

Prof. Julio Cesar Nardi.

Esse tutorial é a Parte 2 da Introdução ao Uso de Serviços Web Usando REST. Portanto, ao seguir este tutorial, você já deve ter feito “Introdução ao Uso de Serviços Web Usando REST - Parte 1”. Vamos usar, inclusive, o projeto Spring Tool Suite já criado quando da realização da Parte 1.

Em especial, neste tutorial, vamos introduzir o uso da ferramenta Postman, muito útil no acesso a serviços web durante a fase de desenvolvimento.

### Usando o mesmo projeto da Parte 1

Como mencionado usaremos o projeto já criado quando da realização da Parte 1 do tutorial Introdução ao Uso de Serviços Web Usando REST. Logo, abra este projeto para iniciarmos nosso trabalho. A estrutura geral de pacotes pode ser observada na Figura 1.

Usaremos as classes *Ator*, *AtorRepository* e *AtorResource* já criadas.

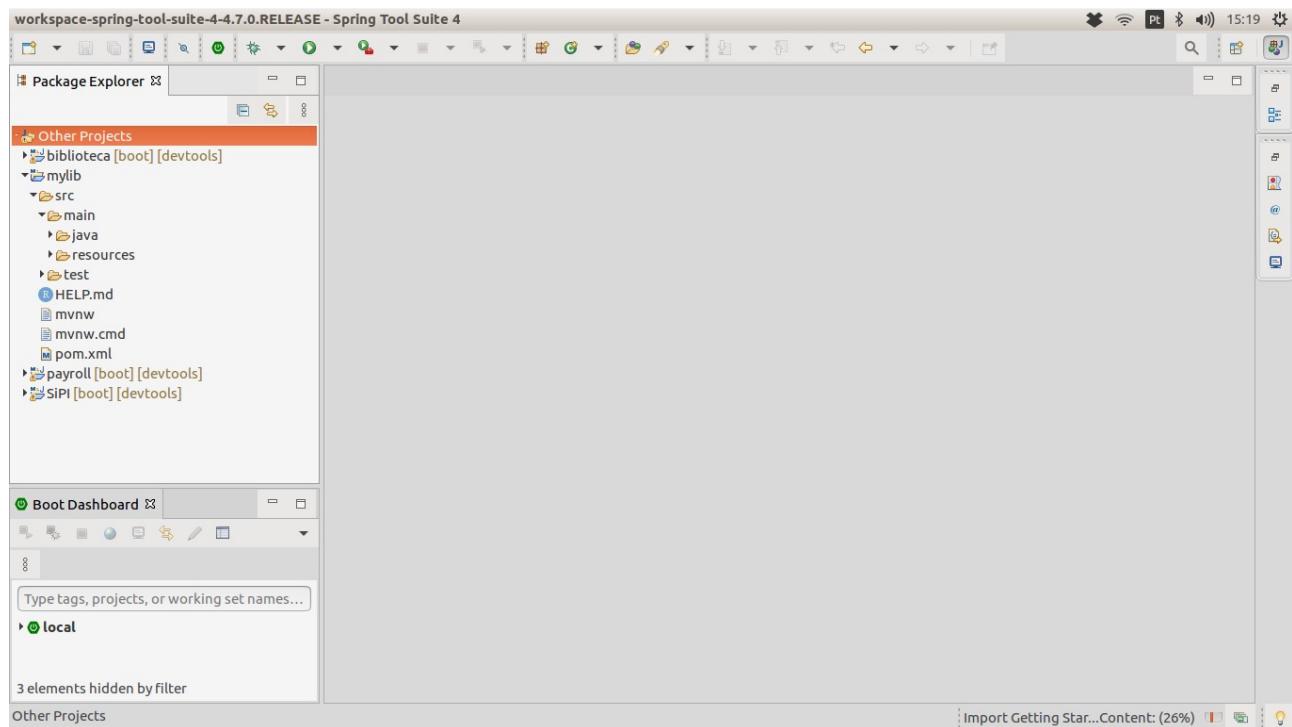


Figura 1: Tela inicial do projeto já criado.

## Criando serviço web de inclusão de objeto no banco de dados

Dentro do pacote `br.edu.ifes.col.mylib.resources`, na classe `AutorResource` vamos criar o método incluir autor, conforme o código-fonte apresentado a seguir.

```
@PostMapping (value="/mylib/resources/ator/{aux_nome}/{aux_cpf}")
public ResponseEntity incluirAutor(@PathVariable(value="aux_nome") String nome,
@PathVariable(value="aux_cpf") String cpf) {

    // Criação do objeto de domínio a ser salvo
    Ator a = new Ator();
    a.setNome(nome);
    a.setCpf(cpf);

    // Tratamento de exceção para eventuais problemas ao salvar o objeto
    try {

        this.repositorioAtores.save(a);

        // Retorno no caso de sucesso.
        return new ResponseEntity(HttpStatus.OK);

    }catch (Exception e) {

        // Retorno no caso de problema.
        return new ResponseEntity(HttpStatus.BAD_REQUEST);
    }
}
```

Observe os códigos usados no método:

1. `@PostMapping (value="/mylib/resources/ator/{aux_nome}/{aux_cpf}")` define o método HTTP (POST), a URL do serviço e o padrão de passagem de parâmetros (*path param*).
2. `@PathVariable(value="aux_nome") String nome, @PathVariable(value="aux_cpf") String cpf` define os parâmetros passados na URL e como esses parâmetros serão mapeados nas variáveis do método.
3. `ResponseEntity` define o tipo de retorno que será usado por nós

Para testarmos o serviço, vamos usar o programa Postman, o qual pode ser baixado em <https://www.postman.com/>. A Figura 2 apresenta a tela inicial do programa.

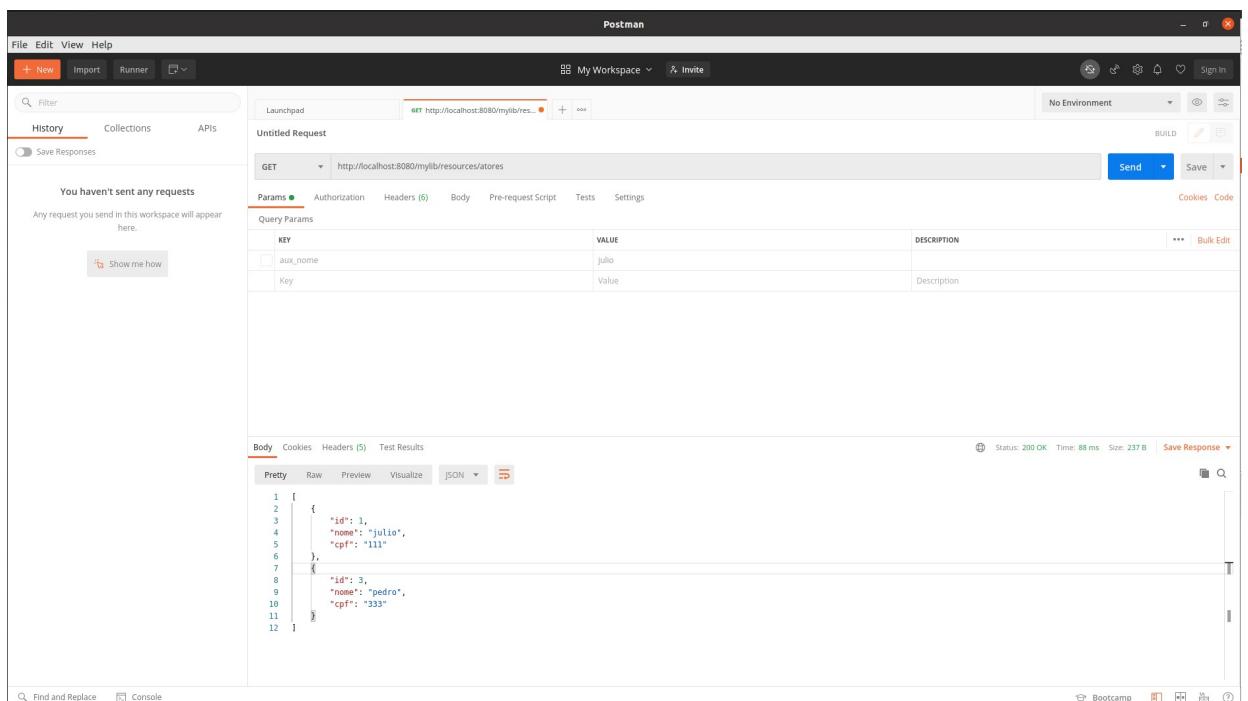


Figura 2: Tela do Postman

A Figura 3 apresenta a tela do programa Postman já com a criação de uma requisição cujo objetivo é testar o serviço criado. Observe que no Request está selecionado o método HTPP a ser utilizado (Post) e a URL do serviço (<http://localhost:8080/mylib/resources/ator/pedro/333>). Clicando em Send, o serviço será acionado e executado e exibindo o “Status: 200 Ok”, no caso de sucesso.

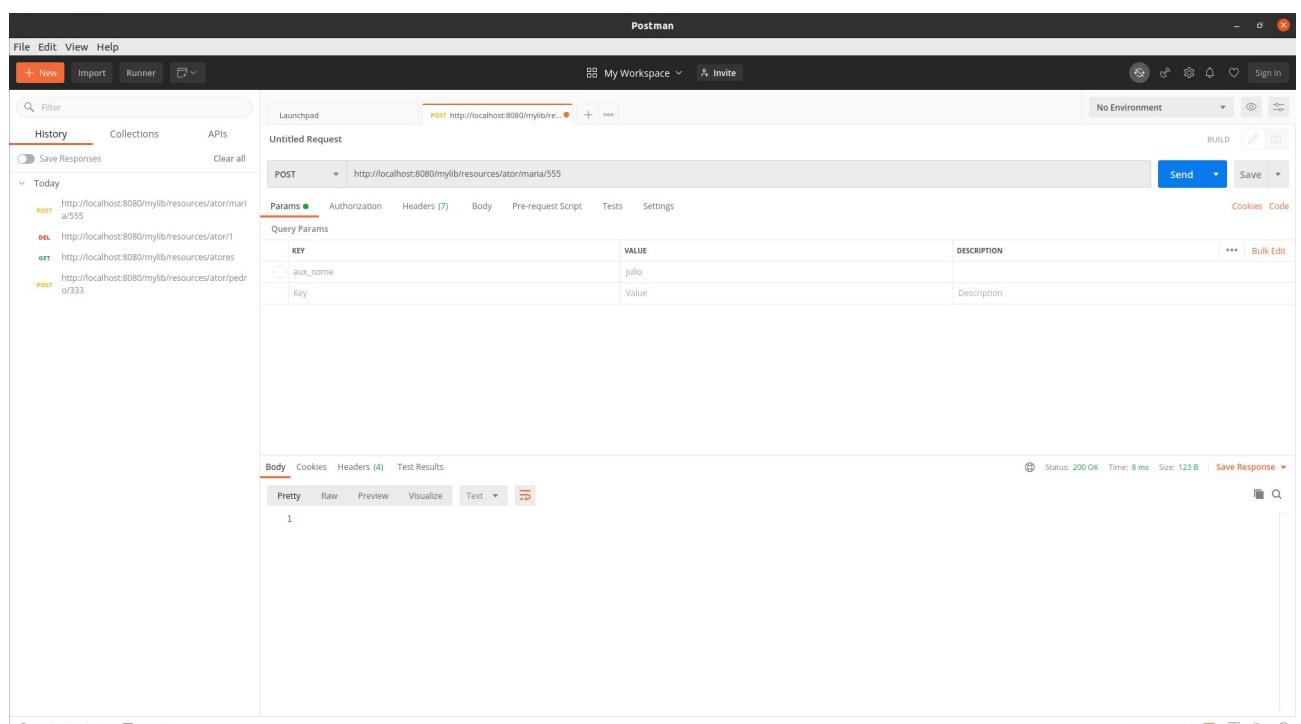


Figura 3: Tela do Postman após ser acionado o serviço de incluir ator.

Para realizar a listagem dos atores já incluídos, vamos acionar serviço criado na Parte 1 do tutorial, ou seja, o serviço que lista todos os atores cadastrados no banco de dados. Para testar esse serviço via Postman, selecione o método HTTP GET, insira a URL correspondente e clique no botão SEND. A Figura 4 apresenta o resultado. Observe que o retorno do serviço aparece na tela do Postman em formato JSON juntamente com o “Status: 200 Ok”.

The screenshot shows the Postman interface with the following details:

- Header Bar:** File, Edit, View, Help, + New, Import, Runner, My Workspace, Invite, No Environment, Sign In.
- Left Sidebar:** History (selected), Collections, APIs, Save Responses, Clear all. History section shows several recent requests including GET, POST, and DELETE operations.
- Central Area:**
  - Request Type:** GET, URL: http://localhost:8080/mylib/resources/atores
  - Params:** aux\_name (Value: julio)
  - Body:** JSON (Pretty) view showing the response body:

```

1  [
2   {
3     "id": 2,
4     "nome": "maria",
5     "cpf": "555"
6   }
7 ]

```
  - Status:** Status: 200 OK, Time: 17 ms, Size: 201 B, Save Response.
- Bottom:** Find and Replace, Console, Bootcamp, Help.

Figura 4: Tela do Postman com a execução do serviço obter atores.

## Criando serviço web de alteração de objeto no banco de dados

Crie o método para alterar ator conforme o código-fonte a seguir.

Por analogia, ajustamos a anotação para **@PutMapping**, a qual faz referência ao método HTTP utilizado e indicamos o formato da URL correspondente. Assim, passando o “ID” do objeto a ser alterado juntamente com os novos dados de “nome” e “cpf” como parâmetro, o serviço localiza o objeto a ser alterado, atribui os novos dados e salva o objeto no banco. Em seguida, realiza o retorno HTTP correspondente a sucesso ou erro. A Figura 6 apresenta a tela do Postman acessando a URL de alteração de ator e o resultado correspondente: “Status: 200 Ok”.

```

@PutMapping (value="/mylib/resources/ator/{aux_id}/{aux_nome}/{aux_cpf}")
    public ResponseEntity alterarAotor(@PathVariable(value="aux_id") Long id,
@PathVariable(value="aux_nome") String nome, @PathVariable(value="aux_cpf") String cpf) {

    // Tratamento de exceção para eventuais problemas ao salvar o objeto
    try {

        Ator a = this.repositoryAtores.findById(id).orElse(null);

        a.setCpf(cpf);
        a.setNome(nome);

        this.repositoryAtores.save(a);

        // Retorno no caso de sucesso.
        return new ResponseEntity(HttpStatus.OK);

    }catch (Exception e) {

        // Retorno no caso de problema.
        return new ResponseEntity(HttpStatus.BAD_REQUEST);
    }

}

```

Para testar, crie dois atores: “Julio”, com cpf “111” e “Maria” com cpf “222”. Em seguida, utilize o serviço de alteração de objetos e altere Julio para: “Sandro” com cpf “777”. A Figura 5 apresenta o resultado da alteração do objeto com “Status: 200 Ok”.

The screenshot shows the Postman application interface. In the top navigation bar, 'Postman' is visible along with standard file operations like File, Edit, View, Help, and a '+' button for creating new requests. Below the header, there's a search bar and a 'Launchpad' section with a 'GET http://localhost:8080/mylib/resources/atores' entry. The main workspace is titled 'Untitled Request' and contains a 'Params' tab where 'aux\_name' is set to 'Julio'. The 'Body' tab shows a JSON payload with two actors: one with id 1 and name 'sandro' (cpf 777), and another with id 2 and name 'maria' (cpf 222). The status bar at the bottom indicates a 'Status: 200 OK' response.

Figura 5: Tela do Postman com o resultado da alteração de objeto.

## Criando serviço web de exclusão de objeto no banco de dados

Agora, vamos criar um serviço para excluir um ator do banco de dados. Veja o código-fonte a seguir.

```
@DeleteMapping (value="/mylib/resources/ator/{aux_id}")
public ResponseEntity excluirAtor(@PathVariable(value="aux_id") Long id) {

    // Tratamento de exceção para eventuais problemas ao salvar o objeto
    try {

        this.repositoryAtores.deleteById(id);

        // Retorno no caso de sucesso.
        return new ResponseEntity(HttpStatus.OK);

    }catch (Exception e) {

        // Retorno no caso de problema.
        return new ResponseEntity(HttpStatus.BAD_REQUEST);
    }
}
```

Por analogia, ajustamos a anotação `@DeleteMapping`, a qual faz referência ao método HTTP utilizado e indicamos o formato da URL correspondente. Assim, passando o ID do objeto como parâmetro, o serviço exclui o objeto do banco de dados e realiza o retorno HTTP correspondente a sucesso ou erro. A Figura 6 apresenta a tela do Postman acessando a URL de exclusão de ator e o resultado correspondente: “Status: 200 Ok”.

The screenshot shows the Postman application interface. In the top navigation bar, 'Postman' is selected. The main area displays an 'Untitled Request' with a 'DELETE' method and the URL 'http://localhost:8080/mylib/resources/ator/1'. Under the 'Params' tab, there are two entries: 'aux\_nome' with value 'julio' and 'Key' with value 'Value'. Below the request details, the 'Body' tab is selected, showing a single character '1'. At the bottom right, the status bar indicates 'Status: 200 OK'.

Figura 6: Tela do Postman exibindo o resultado do serviço de excluir ator.