



Desenvolvimento de Serviços com Spring Boot

Etapa/Aula 02.1

Professor(a): Flávio Neves

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

Roteiro da Aula

- Desenvolver serviços REST básicos usando Spring Boot;
 - Definição do Modelo REST;
 - Funcionamento do Modelo REST;
- Criar aplicações usando método GET através da anotação @RequestMapping;
 - Exemplos de uso do Modelo REST

APIs REST com Spring Boot

- APIs REST são uma forma padronizada de integrar sistemas onde uma aplicação oferece um serviço HTTP (o servidor) e a outra o consome (cliente).
- É um método de comunicação entre máquinas (computador-computador) em uma rede.
- APIs REST foram criadas com o objetivo de possibilitar a interoperabilidade entre os sistemas.

APIs REST com Spring Boot

- O modelo REST (Representational State Transfer - Transferência de Estado Representacional) é uma outra abordagem, uma alternativa ao SOAP (Simple Object Access Protocol).
- O REST é mais que um protocolo, é um estilo arquitetural.
- Ele foi proposto em 2000 por um pesquisador da University of California, Irvine chamado Roy Fielding.
- Trata-se de um padrão aberto, não proprietário.

APIs REST com Spring Boot



APIs REST com Spring Boot

- As principais vantagens do REST são a sua simplicidade e baixo overhead de comunicação.
- Outra característica importante é que ele permite a alta escalabilidade.
- Alta escalabilidade é a capacidade de um sistema escalar, isto é, manter a boa performance mesmo quando o número de requisições aumenta muito.

APIs REST com Spring Boot- Uniform Contract

- A tecnologia REST trabalha com o conceito de uniform contract.
- Isso significa que não é necessário definir contratos individuais para cada web service.
- Os serviços compartilham um contrato uniforme e este contrato é definido por meio dos métodos HTTP.
- Para invocar um serviço REST, basta passar o identificador de recurso (resource identifier).

APIs REST com Spring Boot- Uniform Contract



APIs REST com Spring Boot

- O identificador de recurso é o simples caminho URI (ou URL) onde o serviço se encontra.
- Por exemplo, imagine que você tem um banco de dados de questões de concurso e deseja oferecer um serviço REST com as questões.
- Um identificador de recurso possível seria <http://www.meusite.com.br/questoes/id/123456>.
- Esse é um dos motivos que faz o REST ser mais simples que o SOAP.

APIs REST com Spring Boot

- As comunicação com web services REST, por outro lado, é bem enxuta.
- Na verdade, nada impede que um web service REST se comunique por meio de mensagens XML.
- Mas, quase sempre, web services RESTful recebem e transmitem mensagens no formato JSON.

APIs REST com Spring Boot

- O JSON (JavaScript Object Notation) é um formato auto descritivo mais leve que o XML.
- Veja a seguir um exemplo de comparativo entre 2 documentos: um XML versus um JSON.

APIs REST com Spring Boot

XML

```
<empinfo>
  <employees>
    <employee>
      <name>James Kirk</name>
      <age>40</age>
    </employee>
    <employee>
      <name>Jean-Luc Picard</name>
      <age>45</age>
    </employee>
    <employee>
      <name>Wesley Crusher</name>
      <age>27</age>
    </employee>
  </employees>
</empinfo>
```

JSON

```
{ "empinfo" :
  {
    "employees": [
      {
        "name" : "James Kirk",
        "age" : 40,
      },
      {
        "name" : "Jean-Luc Picard",
        "age" : 45,
      },
      {
        "name" : "Wesley Crusher",
        "age" : 27,
      }
    ]
  }
}
```

APIs REST com Spring Boot

- Note como a mensagem JSON consome bem menos bytes para transmitir as mesmas informações.
- Esse comparativo resume bem 2 vantagens do REST:
 - Simplicidade e
 - Baixo overhead.

APIs REST com Spring Boot

- A forma de trabalhar no REST é por meio de uma série de métodos HTTP que permitem criação, atualização, deleção e consulta de dados.

Métodos HTTP

a idempotência é a propriedade que algumas operações têm de poderem ser aplicadas várias vezes sem que o valor do resultado se altere após a aplicação inicial.

Método	Descrição
GET	Recupera uma entidade em resposta à solicitação de um recurso.
POST	Permite a criação de novos recursos. Retorna uma entidade descrevendo o resultado da ação (novo recurso).
PUT	Atualiza uma entidade (cria a entidade caso ainda não exista).
DELETE	Remove uma entidade.
HEAD	Mecanismo usado por um cliente para verificar a existência de um recurso e, possivelmente, descobrir metadados sobre ele. Não recupera <u>dados</u> do recurso.
PATCH	Permite a realização de <u>atualizações parciais</u> .
OPTIONS	Verifica, no servidor, que outros verbos são aplicáveis a um recurso. Permite que desenvolvedores entendam como interagir com recursos.

APIs REST com Spring Boot

- Um método idempotente é aquele que podemos chamar diversas vezes com a segurança de que todos os retornos serão iguais.
- Não importa se o método é chamado apenas 1 ou 10 vezes. O resultado deve ser o mesmo.
- Essencialmente, significa que o resultado de uma solicitação executada com sucesso é independente do número de vezes que ela é executada.
- No contexto dos serviços REST, os métodos GET, PUT, DELETE, HEAD e OPTIONS são idempotênte.
- Por outro lado, os métodos POST e PATCH não o são.

APIs REST com Spring Boot

- porque o método POST não é idempotente. Ora, a função do POST é justamente criar um recurso, um dado novo.
- Logo, a cada requisição POST, um novo recurso será criado, o que altera o estado da aplicação, fazendo com que seu retorno mude ao longo do tempo.

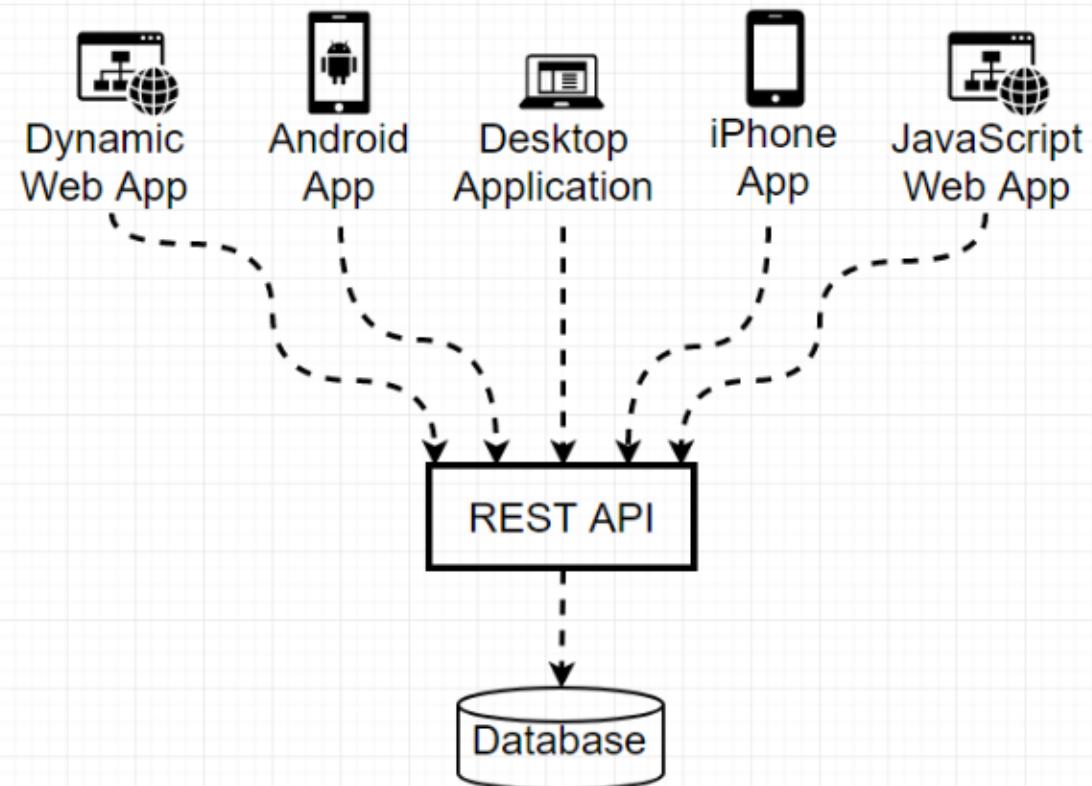
Códigos de status HTTP

- Uma requisição **bem-sucedida** que retorna conteúdo deve gerar o código **200 OK**, enquanto uma **criação com sucesso** responde com **201 Created**.
- O código **204 No Content** é usado quando a ação foi realizada, mas não há dados para retornar.
- Por outro lado, **erros do cliente** resultam em códigos como **400 Bad Request** (requisição malformada) ou **404 Not Found** (recurso inexistente).
- Erros no servidor são representados por códigos da **série 500**, como **500 Internal Server Error**.

Nomenclatura dos Endpoints

- A estrutura de uma API RESTful também deve considerar aspectos como consistência na nomenclatura dos endpoints, evitando verbos nas URLs e utilizando nomes de recursos no plural.
- Por exemplo, em vez de usar /criarUsuario, o ideal é usar POST /usuarios.
- A padronização contribui para uma interface uniforme, facilitando a adoção da API por desenvolvedores externos.

APIs REST com Spring Boot



APIs REST com Spring Boot

- <https://jsonplaceholder.typicode.com/todos/1>
- <https://economia.awesomeapi.com.br/json/last/USD-BRL>
- <https://economia.awesomeapi.com.br/json/last/USD-BRL>
- <https://viacep.com.br/ws/01001000/json/>



Desenvolvimento de Serviços com Spring Boot

Etapa/Aula 02.1

Professor(a): Flávio Neves

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