08

O que é REST

Transcrição

[00:00] Continuando nosso treinamento de API REST com Spring Boot. No vídeo de hoje vou falar um pouco sobre esse modelo, chamado de REST. Esse vídeo é opcional, se você já conhece o modelo ou não tem interesse em conhecer, você pode pular para a próxima aula, porque nele não vamos adicionar nada à nossa API.

[00:19] Como vamos construir uma API REST, é importante conhecermos alguns conceitos desse modelo. "*REST*" é uma abreviação para Representational State Transfer (Transferência do Estado Representacional). É um nome meio esquisito, mas nós vamos entender qual a ideia desse modelo.

[00:40] Ele nada mais é do que um modelo de arquitetura para sistemas distribuídos. Toda a arquitetura que você tem em um sistema conversando com outro é um sistema distribuído. O criador desse modelo REST, o Roy Fielding - que foi um dos criadores do protocolo HTTP - no ano 2000, escreveu sua tese de doutorado, onde ele citava alguns dos modelos que poderiam ser utilizados para arquitetura de sistemas distribuídos. Dentre esses modelos estava o REST.

[01:06] A ideia dele é: se tenho dois sistemas conversando, com transferência de informações via rede, como eu poderia projetar esse sistema para ter uma boa performance, escalabilidade, sem evitar alguns probleminhas?

[01:20] Ele foi a base para a evolução do protocolo HTTP. Hoje a web é toda baseada nesse modelo REST. Depois de muitos anos, o pessoal percebeu que poderia usar esse modelo para fazer integração de sistemas pela web. Quando iam fazer integração via web, utilizavam aquele webservice no modelo SOAP, só que ele era todo baseado em XML, um pouco complexo, e o pessoal começou a pensar que poderiam usar o próprio REST, já que tinham uma arquitetura distribuída, baseada na web, no protocolo HTTP, e queriam ter flexibilidade. Eles começaram a montar APIs seguindo o modelo REST para fazer integração entre sistemas web.

[02:00] Quando falamos de uma aplicação web, o que podemos aproveitar do modelo REST? Existem alguns conceitos importantes que podemos trazer para a nossa API. O primeiro conceito é o de recursos.

[02:20] Toda aplicação gerencia coisas. Se pensarmos no nosso fórum, ele gerencia alunos, tópicos, respostas, cursos, matrículas, entre outras informações. Essas coisas que o sistema gerencia, que uma aplicação gerencia, no modelo REST são chamadas de "recursos". No nosso caso, esses seriam os recursos da nossa aplicação: Aluno, Topico, Resposta, Curso.

[02:51] Como eu tenho vários recursos que a aplicação vai manipular, preciso, de alguma maneira, diferenciar um do outro. Para isso, utilizamos o conceito de URI. Ou seja, cada recurso vai ter uma "*URI*", um Identificador Único. Por exemplo: do aluno, poderia ser "/alunos"; tópicos, "/topicos"; resposta, "/respostas"; curso, "/cursos". Essa URI é o identificador único de cada recurso. Com isso, consigo diferenciar qual recurso que a API vai manipular.

[03:27] Mesmo assim, ainda tenho um problema. Imagine que estou disparando uma requisição para a URI "/alunos". Ou seja, o cliente quer que o servidor manipule o recurso de "aluno" dele. Suponhamos que o servidor perguntasse: "o que você quer fazer nesse recurso?". Ele responderia que pode fazer várias manipulações, como devolver todos os recursos, criar um novo recurso, atualizar,

excluir. É necessário diferenciar qual a operação, como vamos manipular o recurso.

[03:53] Para tratar dessa parte de manipulação, entra outro conceito do HTTP, que são os verbos, os métodos do HTTP. Por exemplo, podemos usar o "GET" para fazer leitura. Se eu disparar uma requisição "GET" para o recurso "/alunos", é porque quero recuperar os recursos. Se eu disparar um "POST", quero cadastrar. O "PUT", atualizar e o "DELETE", excluir.

[04:16] Pelos verbos HTTP consigo diferenciar a manipulação que quero fazer em cima do recurso. Só que, como essa manipulação vai funcionar? Como o cliente me manda uma representação desse recurso? E como a API REST devolve essa representação?

[04:33] É aí que entra o "R" do REST (Representational). A API está transferindo uma representação do recurso. Ela recebe e devolve representações de determinados recursos. E essas representações são feitas pelos "*Media types*", pelos formatos. Geralmente o pessoal utiliza o JSON, mas poderia ser qualquer formato, como XML, HTML, TXT, binário ou qualquer outro.

JSON

```
"aluno":{
    "nome"."Fulano",
        "email"."fulano@email.com"

        }
```

COPIAR CÓDIGO

XML

```
<aluno>
<nome>Fulano</nome>
<email>fulano@email.com</email>
```

COPIAR CÓDIGO

Aqui, por exemplo, temos a representação do recurso "aluno" no formato JSON e outra representação do mesmo recurso, mas num outro media type, o XML.

[05:10] Com isso, consigo representar esses meus recursos. Daí que veio a ideia do nome REST, porque o que o servidor faz é transferir uma representação de um recurso - do estado atual daquele recurso. No exemplo acima, temos o recurso "aluno" que está em determinado estado, com determinado nome e determinado email. Isto é, temos o estado do aluno, uma representação no formato JSON e ela é transferida entre o cliente e o servidor.

[05:38] Perceba que, no modelo REST, a ideia é utilizar os conceitos do protocolo HTTP: as URIs, verbos, representações, media types, entre outros.

[05:54] Esse foi só um pedacinho do REST, só os principais conceitos. Também existem outros importantes, como por exemplo comunicação stateless. A web é baseada no protocolo HTTP, que segue o modelo REST. Como a web consegue suportar bilhões de usuários conectados ao mesmo tempo? Um dos segredos é a "Comunicação Stateless". Os servidores da internet no mundo não estão guardando estado da comunicação. Ela é toda stateless. Não tenho que ter zilhões de servidores com zilhões de gigabytes de memória para guardar estado.

[06:29] A ideia da API REST é também seguir esse modelo de comunicação stateless, sem usar as seções para armazenar dados dos usuários logados.

[06:40] Essa foi só uma breve introdução sobre os principais conceitos do modelo REST. Já que vamos construir uma API REST, precisamos ter um pouco de entendimento sobre esses conceitos, porque vamos utilizar isso na construção da nossa API. Esse era o assunto deste vídeo. Espero vocês no próximo.