REST API

Bootcamp Python
/código[S]
stone

Maicon Francisco de Carvalho
HOW Bootcamps

REST

REST significa REpresentational State Transfer, e é um estilo de desenvolvimento de web services que teve origem na tese de doutorado de *Roy Fielding*. Este, por sua vez, é co-autor do protocolo HTTP. Assim, é notável que o protocolo REST é guiado (dentre outros preceitos) pelo que seriam as boas práticas de uso de HTTP.

Tese de doutorado de Roy T. Fielding

“A vida é um sistema de objetos distribuídos. No entanto, a comunicação entre os humanos é um sistema hipermídia distribuído, onde o intelecto da mente, voz + gestos, olhos + ouvidos e imaginação são todos componentes.”

--- Roy T. Fielding, 1998.

https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

URL’s e URI’s

Um URI pode ser classificado como um localizador, um nome ou ambos. O termo "*Uniform Resource Locator*" (URL) refere-se ao subconjunto de URIs que, além de identificar um recurso, fornecem um meio de localizar o recurso, descrevendo seu mecanismo de acesso primário (por exemplo, sua "localização" de rede).

Recursos

O termo "recurso" é usado em um sentido geral para tudo o que pode ser identificado por uma URI. Exemplos familiares incluem um documento eletrônico, uma imagem, uma fonte de informação com um propósito consistente, um serviço e uma lista de outros recursos.

Por exemplo, seres humanos, empresas e livros encadernados em uma biblioteca também podem ser recursos. Da mesma forma, conceitos abstratos podem ser recursos, como os operadores e operandos de uma equação matemática, os tipos de relacionamento (por exemplo, "pai" ou "funcionário"), ou valores numéricos (por exemplo, zero um e infinito).

Semânticas de recursos

A definição de recurso é baseada em uma premissa simples: as URI devem mudar com a menor frequência possível.

O REST faz isso definindo um recurso como a semântica do que o desenvolvedor pretende identificar.

Resta então ao desenvolvedor garantir que a URI escolhida para uma referência realmente identifique a semântica pretendida.

Exemplos de recursos

http://<dominio>/produtos

http://<dominio>/clientes

http://<dominio>/clientes/5

http://<dominio>/vendas

http://<dominio>/carros

Representações

Em aplicações Web, a representação mais utilizada é o HTML. Essa representação é utilizada como forma de encapsular as informações relacionadas a um determinado recurso.

Além do HTML, podemos utilizar XML, JSON, ou algum outro formato que melhor atenda o cenário que estamos desenvolvendo.

Path parameters

Os path parameters são partes variáveis de uma URI. Eles geralmente são usados para apontar para um recurso específico dentro de uma coleção, como um usuário identificado por ID. Uma URI pode ter vários *path parameters*.

Exemplo:

GET /usuario/{id}

GET /carro/{carro_id}/motorista/{motorista_id}

Query parameters

Os query parameters são passados na própria URL da requisição.

Os query parameters são inseridos a partir do sinal de interrogação. Este sinal indica para o protocolo HTTP que, de alí em diante, serão utilizados query parameters. Esses são inseridos com formato <chave>=<valor> separados com o símbolo &.

Exemplo:

http://<dominio>/carros?marca=vw&modelo=gol

Header parameters

Uma chamada de API pode exigir que os cabeçalhos personalizados sejam enviados com uma solicitação HTTP.

Por exemplo, suponha que uma chamada para GET /ping requer o X-Request-ID header:

Exemplo:

GET /ping HTTP/1.1

Host: example.com

X-Request-ID: 77e1c83b-7bb0-437b-bc50-a7a58e5660ac

Cookie parameters

As operações também podem passar parâmetros no Cookie header, como Cookie: name=value. Vários parâmetros de cookie são enviados no mesmo cabeçalho, separados por ponto e vírgula e espaço.

Exemplo:

```
GET /api/users
Host: example.com
Cookie: debug=0; csrftoken=BUSe35dohU3O1MZvDCUOJ
```

HATEOAS

Hypermedia As the Engine Of Application State é um modelo simples e elegante com que uma API REST pode ser desenhada para que a mesma permita que aplicações que a consumam navegue entre seus recursos através de links e URLs sem a necessidade de conhecimento denso prévio sobre a API.

Isso faz com que a API consiga ser autoexplicativa, guiando o consumidor através de suas requisições.

Exemplo de HATEOAS

```
{
   "cursos": [
      {
         "id": 1,
         "nome": "C# (C Sharp)",
         "aulas": "api.up.com.br/cursos/1/aulas"
      },
      {
         "id": 2,
         "nome": "Python",
         "aulas": "api.up.com.br/cursos/2/aulas"
      }
   ]
}
```

python

Obrigado!

stone + howtech