Desenvolvimento de Software para WEB

Aula 11 - REST

Professor: Anderson Almada

Introdução

- Representational State Transfer ou REST é um modelo de arquitetura (Roy Fielding)

- Muitos desenvolvedores perceberam que também poderiam utilizar o modelo REST para a implementação de Web Services

- REST na verdade pode ser considerado como um conjunto de princípios, que quando aplicados de maneira correta em uma aplicação, a beneficia com a arquitetura e padrões da própria Web.

Identificação de Recursos

- Toda aplicação gerencia algumas informações. (produtos, clientes, vendas),
- Essas informações são os Recursos no modelo REST.
- Um recurso nada mais é do que uma abstração sobre um determinado tipo de informação que uma aplicação gerencia
- Um dos princípios afirma que todo recurso deve possuir uma identificação única.

Identificação de Recursos

- A identificação do recurso deve ser feita utilizando-se o conceito de URI (Uniform Resource Identifier), que é um dos padrões utilizados pela Web.

- Alguns exemplos de URI's:
  - http://servicorest.com.br/produtos
  - http://servicorest.com.br/clientes
  - http://servicorest.com.br/clientes/57
  - http://servicorest.com.br/vendas

URI legíveis

- Utilize nomes legíveis por humanos, que sejam de fácil dedução e que estejam relacionados com o domínio da aplicação.

URI padrão

- Mantenha a consistência na definição das URI’s.

- Crie um padrão de nomenclatura para as URI’s dos recursos e utilize sempre esse mesmo padrão. Evite situações como:
  - http://servicorest.com.br/produto (Singular)
  - http://servicorest.com.br/clientes (Plural)
  - http://servicorest.com.br/processosAdministrativos (Camel Case)
  - http://servicorest.com.br/processos_judidicais (Snake Case).

Evitar operações na URI

- Evite adicionar na URI a operação a ser realizada no recurso

- Os recursos que uma aplicação gerencia podem ser manipulados de diversas maneiras (criar, listar, excluir, atualizar, etc).

- A manipulação dos recursos deve ser feita utilizando-se os métodos do protocolo HTTP

Evitar operações na URI

- Portanto, evite definir URI’s que contenham a operação a ser realizada em um recurso, tais como:

  - http://servicorest.com.br/produtos/cadastrar
  - http://servicorest.com.br/clientes/10/excluir
  - http://servicorest.com.br/vendas/34/atualizar

Evitar alteração de URI

- A URI é a porta de entrada de um serviço. Se você a altera, isso certamente causará impacto nos clientes que estavam a utilizando, pois você alterou a forma de acesso a ele.

- Após definir uma URI e disponibilizar a manipulação de um recurso por ela, evite ao máximo sua alteração.

- Existe a possibilidade de se manter a URI antiga, fazendo um redirecionamento para a nova URI.

Métodos HTTP para operações

- Os recursos gerenciados por uma aplicação, e identificados unicamente por meio de sua URI, geralmente podem ser manipulados de diversas maneiras.

- Quando um cliente dispara uma requisição HTTP para um serviço, além da URI que identifica quais recursos ele pretende manipular, é necessário que ele também informe o tipo de manipulação que deseja realizar no recurso.

- É justamente aí que entra um outro conceito da Web, que são os métodos do protocolo HTTP.

Métodos HTTP para operações

- O protocolo HTTP possui diversos métodos, sendo que cada um possui uma semântica distinta, e devem ser utilizados para indicar o tipo de manipulação a ser realizada em um determinado recurso. V

- Os principais métodos do protocolo HTTP e o cenário de utilização de cada um deles:

Métodos HTTP para operações

GET	Obter os dados de um recurso.
POST	Criar um novo recurso.
PUT	Substituir os dados de um determinado recurso.
DELETE	Excluir um determinado recurso.

Recurso Cliente - Operações HTTP

Método	URI	Utilização
GET	/clientes	Recuperar os dados de todos os clientes.
GET	/clientes/id	Recuperar os dados de um determinado cliente.
POST	/clientes	Criar um novo cliente.
PUT	/clientes/id	Atualizar os dados de um determinado cliente.
DELETE	/clientes/id	Excluir um determinado cliente.

Representação dos recursos

- Os recursos ficam armazenados pela aplicação que os gerencia.

- Quando são solicitados pelas aplicações clientes, por exemplo em uma requisição do tipo GET, eles não “abandonam” o servidor, como se tivessem sido transferidos para os clientes.

- Na verdade, o que é transferido para a aplicação cliente é apenas uma representação do recurso.

Representação dos recursos

- Um recurso pode ser representado de diversas maneiras (XML, JSON,)
- A comunicação entre as aplicações é feita via transferência de representações dos recursos a serem manipulados.
- Uma representação pode ser também considerada como a indicação do estado atual de determinado recurso.
- Desacoplamento entre o cliente e o servidor (facilita a manutenção)

Representação dos recursos - XML

```xml
<clientes>
    <cliente>
        <id>1</id>
        <nome>TreinaWeb Cursos</nome>
        <idade>10</idade>
    </cliente>
</clientes>
```

Representação dos recursos - JSON

```
"clientes" : [
    {
        "id" : 1,
        "nome" : "TreinaWeb Cursos",
        "idade" : 10
    }
 ]
```

Suporte a diferentes representações

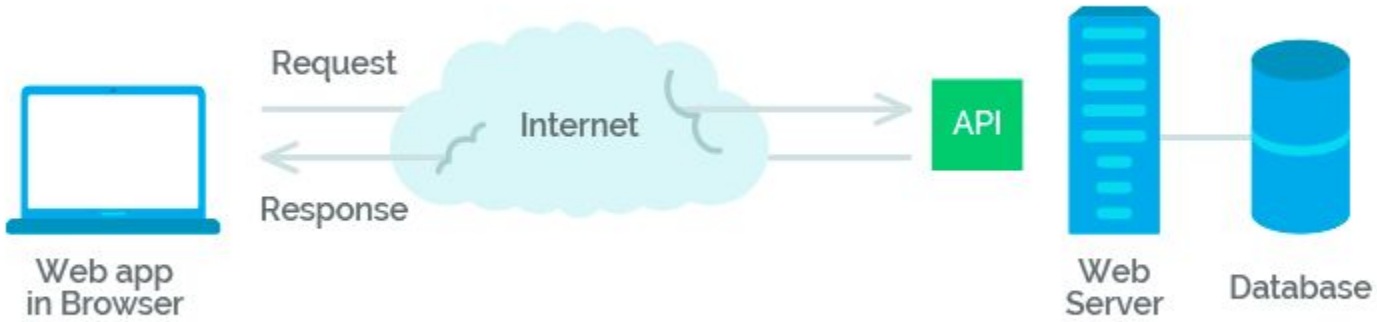
- É considerada uma boa prática o suporte a múltiplas representações em um serviço REST, pois isso facilita a inclusão de novos clientes.

Stateless

- Requisições feitas por um cliente a um serviço REST devem conter todas as informações necessárias para que o servidor as interprete e as execute corretamente.

RESTful

- RESTful é a aplicação ou serviço que seguir todos os princípios do REST



Links importantes

- https://blog.caelum.com.br/rest-principios-e-boas-praticas/
- https://www.treinaweb.com.br/blog/rest-nao-e-simplesmente-retornar-json-indo-alem-com-apis-rest/
- https://www.restapitutorial.com/

Dúvidas??

E-mail: almada@crateus.ufc.br