REST - Representational State Transfer

Caroline de Morais Dias,
Guilherme Luigi Andrade Campachi Arbeolya,
Jessé Rodrigues de Souza

Faculdade de Tecnologia do Estado de São Paulo – FATEC Carapicuíba

Av. Francisco Pignatari – Carapicuíba – SP – Brasil

moraiscaah@gmail.com, guiluigi1712@gmail.com, jesse.rsouza@gmail.com

Abstract. *This paper covers the concept of REST, the history and where can be used. Also, explains what are the resources and methods, URLs and URIs. At the end, we'll see some Java specifications with RESTful web services and what is and how to use JSON.*

Resumo. *Este artigo aborda o conceito do REST, a história e onde pode ser usado. Explica também o que são os recursos e métodos, URLs e URIs. Ao final, veremos um pouco das especificações RESTful com web services em Java e o que é e como usar o JSON.*

1. Introdução

Este artigo tem objetivo de trazer a definição de REST, SOAP e webservices que são consumidas nestes modelos. Traz também a definição do URL e URI, o protocolo HTTP e seus métodos e a definição da API JAX-RS para aplicações REST desenvolvidas em Java.

2. REST e SOAP

Representational State Transfer, mais conhecido como REST, é um modelo utilizado para se projetar arquiteturas de sistemas distribuídos, baseado em comunicação em rede sob o protocolo HTTP/HTTPS.

É possível utilizar diversos formatos de texto, como CSV (Comma-Separed Values), RSS (Really Simple Syndication), JSON (JavaScript Object Notation) e YAML (Yet Another Markup Language) e possui uma curva de aprendizado bem melhor, se comparado ao SOAP.

SOAP (Simple Object Access Protocol) também é uma arquitetura, mas sua tecnologia foi desenvolvida pela Microsoft. Utilizada também como meio de comunicação via HTTP, utiliza o formato XML.
É possível utilizar os métodos HTTP oficiais e também personalizados para efetuar o tráfego de arquivos. Porém, quando utilizado métodos personalizados a aplicação deixa de ser qualifica como REST.

3. HTTP / HTTPS

HTTP é um protocolo de comunicação em rede, estilo cliente-servidor, que permite a obtenção de recursos.

Com o HTTP é possível controlar cache, sessão, autenticação, proxy e tunelamento. Muito utilizado no acesso de websites e base para diversos modelos de webservices.

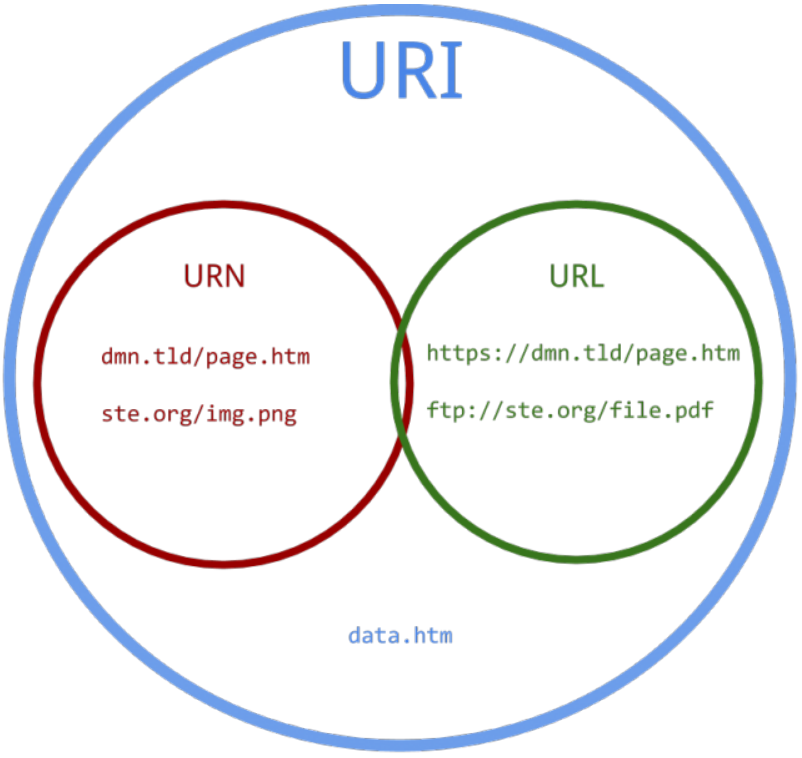
Atualmente existe o HTTP/2, uma versão atualizada do HTTP que surgiu para suprir problemas de latência.

Há também o HTTPS, que adiciona uma camada de segurança que garante a proteção na autenticação e proteção e integridade das informações trafegadas por meio de criptografia.

4. URI e URL (Recursos)

Uniform Resource Identifier (URI) é o endereço, um recurso único utilizado como parâmetro para troca de dados.

URL é um tipo de URI, que é utilizado junto com algum tipo de protocolo de comunicação.



URI, URN e URL

URI também possuí sintaxe :

A sintaxe começa pelo protocolo, onde é definido como o recurso será acessado:

Logo após vem o domínio, podendo ser um endereço IP ou um DNS (Domain Name System):

Depois vem a porta aonde o endpoint será disponibilizado (geralmente não é necessário indicar a porta):

Quando se deseja um arquivo, é indicado um caminho:

Em alguns casos é necessário buscar um valor específico, neste momento indicamos os parâmetros:

Em outros casos não é necessário uma busca por query, apenas indicamos uma âncora para navegar até a parte que desejamos:

5. Métodos

O protocolo HTTP necessita de alguns métodos para poder executar chamadas e realizar a comunicação entre cliente e servidor.
Estes métodos são:

GET: O método mais comum. Por este tipo de requisição é solicitada a representação do recurso, podendo ser .html, .json, .xml, .mp4, .php e etc.

POST: Envia uma entidade e requisita que o recurso seja guardado na URI fornecida.

PUT: Requisita que um entidade seja armazenada na URI fornecida. Se a URI se refere a um recurso que já existe, ele é modificado, se a URI não aponta para um recurso existente, então o servidor pode criar o recurso com essa URI.

PATCH: Usado para aplicar modificações parciais a um recurso.

DELETE: Apaga o recurso especificado.

HEAD: Retorna os cabeçalhos de uma resposta.

TRACE: Devolve a requisição recebida para que o cliente veja se houveram mudanças e adições feitas por servidores intermediários.

OPTIONS: Retorna os métodos HTTP que o servidor suporta para a URL especificada.

CONNECT: Converte a requisição de conexão para um túnel TCP/IP transparente, usualmente para facilitar comunicação criptografada com SSL (HTTPS) através de um proxy HTTP não criptografado.

6. JSON

JavaScript Object Notation é uma forma de representar e trocar informações em texto. Por ser menos verboso que o XML e mais leve, mesmo trafegando muita informação, atualmente é um dos meios mais usados em RESTful APIs.

Um JSON pode ser armazenado em um arquivo .json, possuindo o MIME type: application/json.

7. XML

XML (Extensible Markup Language) é uma linguagem de marcação similar à HTML. Existe uma recomendação pela W3C definindo-a como uma linguagem de marcação de propósitos gerais.

Existem algumas linguagens baseadas em XML: Por exemplo XHTML, MathML, SVG, XUL, XBL, RSS e RDF.

8. Especificação Java para o uso de RESTful Web Services

Desde o Java EE 6 existe o JAX-RS, que é uma API JAVA que, segundo a documentação oficial da Oracle, foi desenvolvida para facilitar aplicações que utilizam arquitetura REST.

O JAX-RS usa *anotations* (que funcionam em tempo de execução) para simplificar o desenvolvimento de aplicações *webservices* RESTful.

Exemplo disponível na documentação da Oracle:

```
package com.sun.jersey.samples.helloworld.resources;

import javax.ws.rs.GET;
import javax.ws.rs.Produces;
import javax.ws.rs.Path;

// A classe Java será hospedada na URI "/helloworld"

@Path("/helloworld")
public class HelloWorldResource {

    // O Java irá processar um método GET
    @GET
   // O Java indica que o conteúdo da requisição será do tipo texto
    @Produces("text/plain")
    public String getClichedMessage() {
        // Retorna um valor do tipo texto (string)
        return "Hello World";
    }
}
```

9. Conclusão

Neste artigo foi identificado o que é REST, SOAP e o uso com webservices. Também foi abordado a definição de HTTP/HTTPS e seus métodos, a definição de JSON e XML.
No final foi abordado a JAX-RS, uma definição Java para lidar com webservices que utilizam arquitetura RESTful.

Referências

FIELDING, Roy (2000) - "Architectural Styles and the Design of Network-based Software Architectures"

http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm, Acesso: Abril/2018

FERREIRA, Rodrigo (2017) - "REST: Princípios e boas práticas"

http://blog.caelum.com.br/rest-principios-e-boas-praticas/, Acesso: Abril/2018

RICHARDSON, Leonard, RUBY, Sam (2007) - "RESTful Web Services"
https://www.crummy.com/writing/RESTful-Web-Services/RESTful_Web_Services.pdf, Acesso: Abril/2018

ARMIGLIATTO, Guilherme Moraes (2017) - “REST usa JSON e SOAP usa XML, certo?”

http://matera.com/br/2017/01/10/rest-usa-json-e-soap-usa-xml-certo/, Acesso: Maio/2018

MOZILLA (2018) - “Uma visão geral do HTTP”

https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview, Acesso: Abril/2018

MOZILLA (2018) - “Indentifying resources on the Web”

https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Basico_sobre_HTTP/Identifying_resources_on_the_Web, Acesso: Abril/2018

MOZILLA (2018) - “Working Whit JSON”

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON, Acesso: Abril/2018

FERREIRA, Gabriel (2015) - “Os métodos de requisição HTTP: quais são e para que servem?”
http://gabsferreira.com/os-metodos-http-e-a-diferenca-entre-eles/, Acesso: Maio/2018

ORACLE (2013) - “Building RESTful Web Services with JAX-RS”
https://docs.oracle.com/javaee/6/tutorial/doc/giepu.html, Acesso: Maio/2018