



Universidade Federal de Uberlândia
Faculdade de Computação

Programação para Internet

Módulo 11

Introdução aos Serviços Web e ao
Desenvolvimento de Aplicações Web com Java

Prof. Dr. Daniel A. Furtado

Introdução aos Serviços Web

Web Services

- **Definição.** *Um **web service** é uma entidade de software baseada em padrões, independente de linguagem, que aceita requisições especialmente formatadas de outras entidades de softwares em máquinas remotas por meio de protocolos de comunicação universais, produzindo respostas específicas da aplicação.*
- De maneira simplista, *web services* são serviços de software que aplicações web disponibilizam para outras aplicações web utilizando padrões de comunicação universais, como HTTP com JSON ou XML;

Ref.: Ioannis G. Baltopoulos

Web Services - Exemplos

- Exemplo de *web service* para busca de informações de endereço a partir do CEP:
 - <https://viacep.com.br/ws/38408100/json/>

Web Services - Benefícios

- **Interoperabilidade e Integração.** Empresas frequentemente possuem sistemas diferentes, com dados próprios. **Web services** facilitam a comunicação e integração desses sistemas (e até mesmo entre sistemas de empresas diferentes).
- **Reusabilidade.** Uma função disponibilizada dentro de um domínio por meio de um *web service* pode ser codificada uma única vez e utilizada inúmeras vezes por outras aplicações. Por exemplo, é possível que um mesmo *web service* seja acessado por uma aplicação interna, um browser em um desktop, um aplicativo de celular ou por um navegador *mobile*.
- **Back-end/Front-end.** *Web services* viabilizam a separação do front-end do back-end em sistemas Web.

Tipos de Serviços Web

Tipos de Web Services

- Os ***web services*** são comumente categorizados de acordo com a tecnologia em que se baseiam:
 - SOAP
 - REST

Tipos de Web Services

SOAP

- Protocolo baseado na **linguagem XML**, especificado pelo W3C, para a troca de mensagens entre aplicações na Internet/Intranet
- Acrônimo para *Simple Object Access Protocol*
- Como é baseado na XML, é independente de plataforma ou linguagem
- Possibilita chamadas a métodos remotos (RPC), inclusive com argumentos complexos, como se fossem chamadas locais;
- Provê uma espécie de “envelope” para o envio de mensagens de serviços web através da Internet/Intranet
- Utiliza documentos **WSDL** (*Web Service Description Language*) para descrever os serviços (localização, métodos, tipos de dados, etc)

SOAP - Exemplo

1) Exemplo de mensagem de requisição SOAP para buscar no servidor a cotação de uma ação na bolsa

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding"

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

2) Exemplo de uma mensagem de resposta SOAP com o preço da ação

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Uma desvantagem clara do SOAP é o overhead com *metadados*, o que demanda uma largura de banda maior.

Adaptado de w3schools.com

Tipos de Web Services

REST

- *Representational State Transfer*;
- **Estilo arquitetural** para comunicação entre aplicações na Web;
- Se baseia, na maioria das vezes, no **protocolo HTTP** e seus **códigos de status** e nos **métodos de requisição** (POST, GET, PUT, PATCH, DELETE, OPTIONS e HEAD);
- Diferente do SOAP, não impõe restrições ao formato da mensagem, mas apenas no comportamento das entidades envolvidas:
 - **Mais flexível**: o desenvolvedor pode utilizar o formato que for mais apropriado, como XML, JSON, texto, etc.
- Trata objetos no servidor como recursos que podem ser criados, modificados ou removidos;
- Independente de linguagem de programação;

RESTful

- Termo comumente utilizado para designar aplicações ou APIs baseadas no estilo REST

Web Service Rest - Exemplos

- O serviço de busca de endereço mostrado anteriormente é um exemplo de serviço RESTful:
 - <https://viacep.com.br/ws/38408100/json/>
- Nesse exemplo, para obter a resposta no formato xml, por exemplo, basta trocar a palavra **json** na URL por **xml**

SOAP vs REST

Leitura recomendada:

<http://www.java2blog.com/2016/06/difference-between-soap-and-rest-web-services.html>

Conceito de Idempotência e Métodos HTTP no Contexto de Serviços RESTful

Web Services e Idempotência

Um requisição HTTP é dita **idempotente** quando mantém a seguinte propriedade:

- *Executar a requisição múltiplas vezes tem o mesmo efeito que executá-la uma única vez*

Principais Métodos HTTP para serviços *RESTful*

POST

- Utilizado para **criar** novos recursos (por exemplo, em conjunto com a operação *INSERT* da SQL)
- Em geral, altera o estado da aplicação no servidor
- Em caso de sucesso, deve-se retornar o código HTTP 201 (Created)
- Por definição, não é idempotente. Isto significa que sucessivas requisições* utilizando o POST podem ter efeitos diferentes (por exemplo, podem resultar na criação de dois ou mais recursos contendo a mesma informação).

*Requisições idênticas

Lista dos códigos de status HTTP: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Mais detalhes sobre POST: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST>

Principais Métodos HTTP para serviços *RESTful*

GET

- Utilizado normalmente para operações de **leitura** de recursos (por exemplo, em conjunto com a operação *SELECT* da SQL);
- Não altera o estado da aplicação no servidor;
- Em caso de sucesso, retorna-se uma representação do recurso no formato *XML*, *JSON* ou *Texto* e o código HTTP 200 (OK)
- Por definição, **é idempotente**. Isto significa que um usuário (ou alguma ferramenta de software) pode executar a mesma requisição inúmeras vezes sem se preocupar em produzir efeitos diversos no servidor.
- Leitura complementar: <https://stackoverflow.com/questions/18395523/what-is-difference-between-http-methods-get-post-put-and-delete>

Mais detalhes sobre o método GET:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET>

Principais Métodos HTTP para serviços *RESTful*

PUT

- Utilizado para operações de **atualização/substituição** por inteiro de um recurso (muitas vezes, envolve a operação *UPDATE* da SQL);
- Semelhante ao método POST, porém com a diferença de **ser idempotente**;
- Em outras palavras, se um recurso é atualizado por meio de uma requisição PUT e, na sequência, a mesma requisição é repetida outras vezes, então todas elas terão o mesmo resultado, uma vez que o recurso atualizado permanecerá no mesmo estado que estava logo após a primeira requisição;
- Em caso de sucesso, os possíveis códigos de status de retorno são 200, 201, 204
- Leitura complementar: <https://stackoverflow.com/questions/23777714/actual-use-of-get-put-delete-post-methods-in-http>

Mais detalhes sobre o método PUT:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT>

Principais Métodos HTTP para serviços *RESTful*

PATCH

- Frequentemente utilizado para operações de **atualização parcial** de um recurso (por exemplo, atualização de um dado em particular de um cliente, como número de telefone ou estado civil);
- Diferente do PUT, **PATCH** não é idempotente, o que significa que requisições sucessivas utilizando PATCH podem ter efeitos diferentes;

Mais detalhes sobre o método PATCH:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH>

Principais Métodos HTTP para serviços *RESTful*

DELETE

- Utilizado para **remover** um recurso (por exemplo, envolvendo a operação *DELETE* da SQL);
- Em caso de sucesso, deve-se retornar o código HTTP 200 (OK)
- **É idempotente.** Repetidas requisições para remoção do mesmo recurso devem ter o mesmo resultado (200 – OK), ou seja, o recurso foi apagado e continua apagado.

Uma descrição completa de todos os métodos HTTP pode ser obtida em:

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>