Web Services - Conceitos básicos

Emilio Francesquini
emilio@ime.usp.br[1]

Janeiro de 2010

Segunda-feira, logo pela manhã...

Serviços - Definição

- Zhang (2007) em uma publicação especializada do IEEE [1] define um serviço como:

  "Um serviço é um tipo de relacionamento (contrato) entre um provedor e um consumidor, sendo que esse provedor se compromete em realizar determinadas tarefas com resultados pré-estabelecidos para um consumidor, e que, por sua vez, se compromete a usar o serviço da forma contratada."

Web Services - Definição

- O W3C define um web service como [2]:

  "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."

Na Web não faltam exemplos...

- RSS
- Google Calendar (CalDAV sobre HTTP) sincronizável com o iCal, por exemplo
- Amazon Web Services (SOAP ou REST sobre HTTP)
  - EC2
  - S3
  - SimpleDB
  - RDS

Web Services

- Muito mais "comuns" em intranets do que na internet
- Tem sido muito utilizados para
  - RPC, RMI, ...
  - Integração de sistemas substituindo *
    - Troca de arquivos zonados
    - Tabelas em BD
    - TCP/IP
    - CORBA, JRMI, ...

Web Services - Vantagens

- Interface simples de ser implementada
- Qualquer linguagem que tenha possibilidade de utilizar soquetes TCP já está virtualmente apta a utilizar um web service
- Por ser sobre HTTP é mais provável que seja capaz de passar por firewalls, proxies e afins
- É o padrao de fato do mercado
- Baixo acoplamento e possibilidade de evolução da interface mantendo compatibilidade com versões anteriores

Web Services - Desvantagens

- Lentidão
- Disparidade muito grande do padrão especificado e do que as pessoas realmente fazem
- Toda integração é sempre uma grande surpresa

Web Services

- As duas maneiras mais comuns para fazer acesso aos web services hoje são:
  - SOAP (Simple Object Access Protocol)
  - REST (REpresentational State Transfer), também chamados de RESTful web services
- Uma das poucas coisas com que se pode contar é que será XML sobre HTTP

HTTP

- Uma escolha natural
- Permitiu um crescimento rápido utilizando a infra-estrutura já existente
- Tecnologia testada e segura com oferecimento de transmissão criptografada caso necessário

XML

- eXtensible Markup Language criada em 1996 pelo W3C como uma simplificação da SGML (Standard Generalized Markup Language)
- Formato legível por humanos
- Fácil para fazer o parsing

XML - Exemplo

```
<?xml version="1.0"/>
<bibliografia>
    <livro disponivel="T"> <!-- isso é um comentário -->
        <titulo>Memórias Póstumas de Brás Cubas</titulo>
        <autor>Machado de Assis</autor>
        <ano>1881</ano>
    </livro>
    <livro disponivel="F">
        <titulo>A Cidade e as Serras</titulo>
        <autor>Eça de Queirós</autor>
        <ano>1901</ano>
    </livro>
</bibliografia>
```

DTD

- Data Type Definition
- Define as tags e os valores válidos a serem utilizados em um XML
- Vale notar que não surgiu como XML e já vinha sendo utilizada como validadora de documentos SGML, tecnologia precursora do XML
- Podem ser internos ou externos a um XML

DTD – Exemplo

```
<!DOCTYPE bibliografia [
  <!ELEMENT bibliografia (livro*)>
  <!ELEMENT livro (titulo, autor+, ano, editora)>
  <!ELEMENT titulo (#PCDATA)>
  <!ELEMENT autor (#PCDATA)>
  <!ELEMENT ano (#PCDATA)>
  <!ELEMENT editora (#PCDATA)>
]>
```

DTD – Exemplo 2

```
<!DOCTYPE jornal [
  <!ELEMENT jornal (artigo+)>
  <!ELEMENT artigo (manchete, noticia, notas+)>
  <!ELEMENT manchete (#PCDATA)>
  <!ELEMENT noticia (#PCDATA)>
  <!ELEMENT notas (#PCDATA)>
  <ATTLIST artigo
  autor CDATA #REQUIRED
  data CDATA #IMPLIED
  edicao CDATA #IMPLIED
  >
]>
```

Ligação XML DTD

- Pode ser interno
  - Basta colocar o DTD logo após o cabeçalho do XML
- Externo
  - Adiciona-se a linha abaixo logo após o cabeçalho do XML

```
<!DOCTYPE nome-tag-raiz SYSTEM
"nomeDoArquivo.dtd">
```

XSD

- O W3C define um schema como um documento responsável por declarar para um XML:
  - Os elementos e seus atributos
  - A hierarquia dos elementos
  - A ordem dos elementos
  - Número de elementos de um elemento complexo
  - Se um elemento é vazio ou se aceita texto
  - Os tipos de dados de elementos e atributos
  - Valores padrão para elementos e atributos

XSD - Exemplo

```
<?xml version="1.0"?>
<disciplina>
  <codigo>MAC110</codigo>
  <nome>Introdução à Computação</nome>
  <ministrante>Chuck Norris</ministrante>
  <creditos>4</creditos>
</disciplina>
```

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ime.usp.br"
xmlns="http://www.ime.usp.br"
elementFormDefault="qualified">
  <xs:element name="disciplina">
    <xs:complexType> <xs:sequence>
      <xs:element name="codigo" type="xs:string"/>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="ministrante" type="xs:string"/>
      <xs:element name="creditos" type="xs:integer"/>
    </xs:sequence> </xs:complexType>
  </xs:element>
</xs:schema>
```

XML com XSD

```
<?xml version="1.0"?>
<ime:disciplina
xmlns:ime="http://www.ime.usp.br"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://www.ime.usp.br">
  <ime:codigo>MAC110</codigo>
  <ime:nome>Introdução à Computação</nome>
  <ime:ministrante>Chuck Norris</ministrante>
  <ime:creditos>4</creditos>
</ime:disciplina>
```

Restrições

```
<xs:element name="nome" type="xs:string">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="10"/>
      <xs:maxLength value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Namespaces

```
<?xml version="1.0"?>

<grade
xmlns:ime="http://www.ime.usp.br/schema"
xmlns:fau="http://www.fau.usp.br/schema">

   <ime:disciplina nome="Introdução à
   Computação"/>

   <fau:disciplina descricao="Introdução ao
   Urbanismo"/>

</grade>
```

SOAP

- Simple Object Acess Protocol
- Usa XML sobre HTTP
- Basicamente um RPC
- Diferentemente de CORBA e JRMI as chamadas não mantém estado
  - Algumas tentativas de definir chamadas com estado foram feitas, sendo a mais promissora WSFR (Web Services Resource Framework) um esforço de IBM e Oracle entre outras
  - Outra forma de pensar

SOAP - WSDL

- As interfaces em SOAP são definidas através da WSDL – Web Service Definition Language
- WSDL é um documento XML que traz operações, parâmetros, retornos, exceções e pontos de acesso
- Pontos de acesso são, no final, as URLs dos serviços

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://server/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://server/" name="HelloWorldService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://server/"
      schemaLocation="http://localhost:8080/hello?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="sayHello">
    <part name="parameters" element="tns:sayHello"/>
  </message>
  <message name="sayHelloResponse">
    <part name="parameters" element="tns:sayHelloResponse"/>
  </message>
  <portType name="HelloWorld">
    <operation name="sayHello">
      <input message="tns:sayHello"/>
      <output message="tns:sayHelloResponse"/>
    </operation>
  </portType>
...
</definitions>
```

```
  …
  <binding name="HelloWorldPortBinding" type="tns:HelloWorld">
    <soap:binding
    transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>
    <operation name="sayHello">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="HelloWorldService">
    <port name="HelloWorldPort"
    binding="tns:HelloWorldPortBinding">
      <soap:address location="http://localhost:8080/hello"/>
    </port>
  </service>
</definitions>
```

UDDI

- Universal Description, Discovery, and Integration
- Padrão para a publicação e localização de web services
- Baseado em SOAP
- Capaz de responder perguntas sobre:
  - Responsável por um web service
  - O que ele faz
  - Onde se localiza
  - Como acessar
- USML

REST

- REpresentational State Transfer
- Serviços que seguem essa linha são comumente chamados de RESTful
- A minha apresentação será pragmática, para uma visão mais teórica consulte a tese de doutoramento de Roy T. Fielding (2000)
- Alguns podem achar essa abordagem prática demais. :)

REST - Princípios

- Atribua identificadores a tudo
- Associe os recursos
- Utilize métodos padrão
- Recursos podem possuir múltiplas representações
- Comunique-se sem manutenção de estado

REST - Identificadores

- Todas as abstrações do sistema, merecem ser identificadas
- Não apenas uma abstração individualmente deve ser etiquetada, mas qualquer conjunto, que faça sentido, também pode ser identificado por um id
  - Todas as latas de ervilha
  - Todas as pessoas que tem endereço em SP

REST - Identificadores

- Na web, o jeito mais natural de fazer tal identificação é através de URIs
  - http://example.com/customers/1234
  - http://example.com/orders/2007/10/776654
  - http://example.com/orders/2007/11
  - http://example.com/products?color=green
- Sem paúra!
  - Anos de prática OO nos ensinaram a não expor os detalhes de implementação, ainda mais Ids no BD
  - O fato é que os conceitos (recursos) que você desejará expor são geralmente muito mais abrangentes do que uma linha no BD

REST - Associação

- HATEOAS

REST - Associação

- HATEOAS - Hypermedia As The Engine Of Application State
- !!!
- Significa que tudo deve ser ligado, imagine o seguinte exemplo de uma resposta a uma requisição de pesquisa por pedidos

```
<pedido ref='http://xyz.com/pedido/1234'>
  <qtd>23</qtd>
  <produto ref='http://abc.com/produtos/4554'/>
  <cliente ref='http://jkl.com/clientes/7654' />
</pedido>
```


Piso Baixo 5 2505
DTB-5009
VIA SUL
Piso Baixo
01 - TIRADENTES
02 - TIRADENTES
SAÍDA
ELEVADOR DE ESCADA

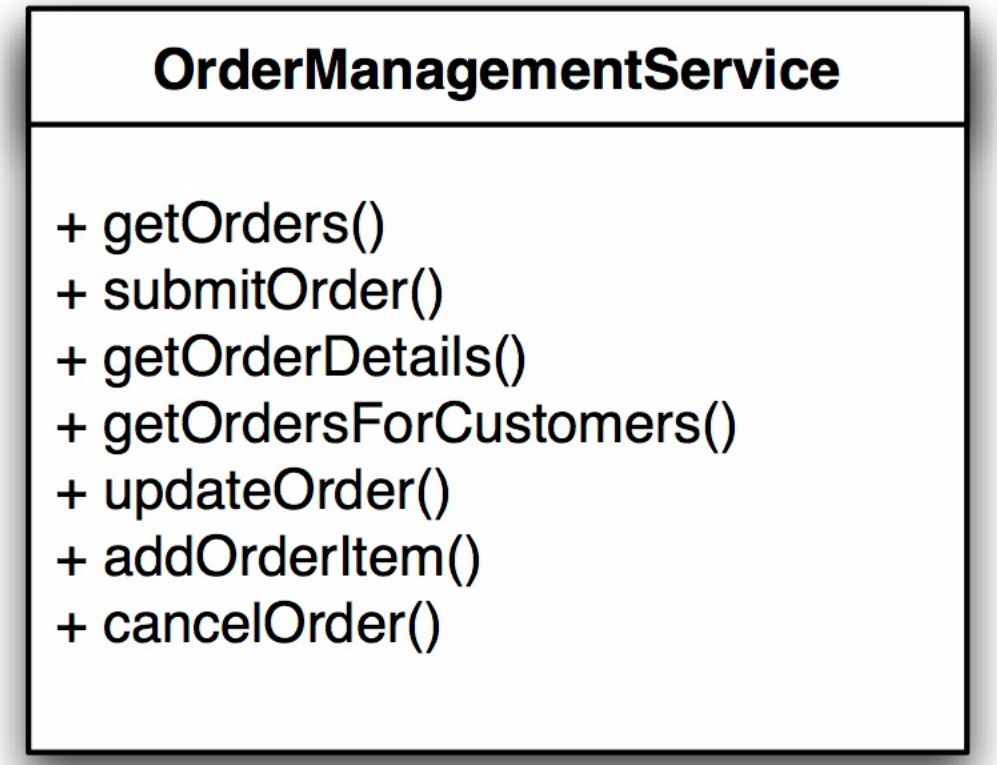
REST - Métodos padrão

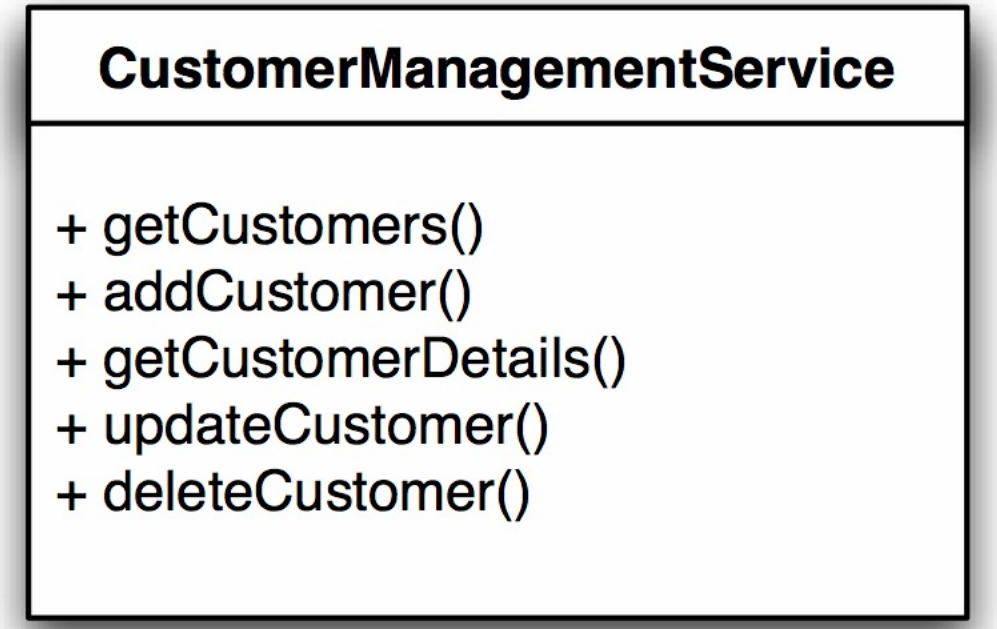
- GET – Pega uma representação do recurso
- PUT – Cria ou atualiza, se já existir o recurso, com as informações passadas
- DELETE – Apaga o recurso
- POST – Única operação não idempotente. Geralmente significa criação de um recurso ou ativação de algum processamento arbitrário, logo também não é seguro
- Pode-se imaginar que todo recurso é algo semelhante a

```
class Resource {
    Resource(URI u);
    Response get();
    Response post(Request r);
    Response put(Request r);
    Response delete();
}
```

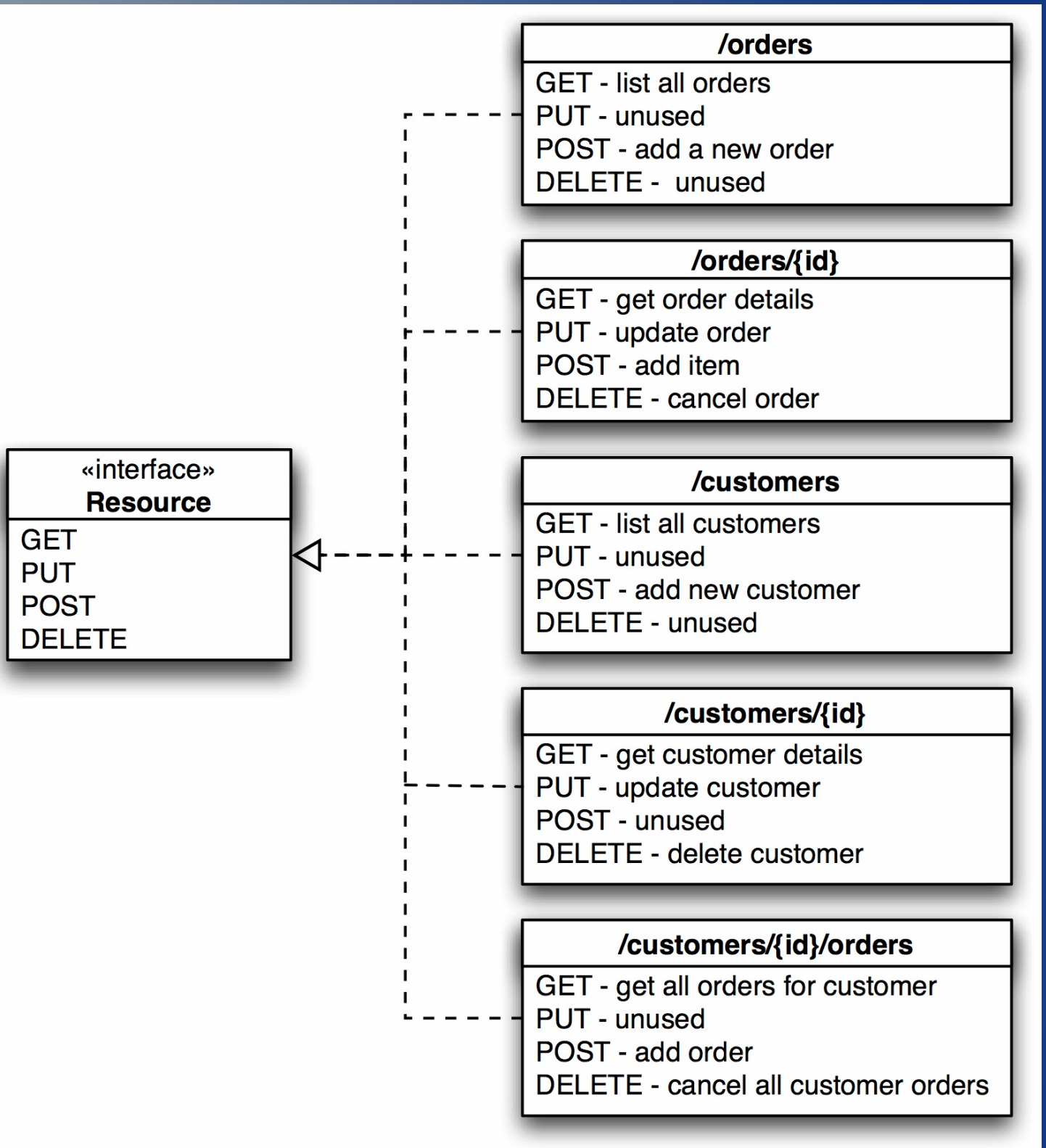
REST – Métodos padrão

- Essas restrições devem ser respeitadas pelos serviços RESTful
- Infelizmente, como quase tudo no mundo dos web services, essas restrições, como são apenas semânticas, acabam sendo ignoradas por grande parte dos serviços disponíveis
- Pode ser difícil imaginar que a sua aplicação modelada de uma maneira OO possa ser mapeada desta maneira





Tilkov, 2007 [3]


«interface»
Resource
GET
PUT
POST
DELETE
/orders
GET - list all orders
PUT - unused
POST - add a new order
DELETE - unused
/orders/{id}
GET - get order details
PUT - update order
POST - add item
DELETE - cancel order
/customers
GET - list all customers
PUT - unused
POST - add new customer
DELETE - unused
/customers/{id}
GET - get customer details
PUT - update customer
POST - unused
DELETE - delete customer
/customers/{id}/orders
GET - get all orders for customer
PUT - unused
POST - add order
DELETE - cancel all customer orders

REST - Múltiplas representações

REST - Múltiplas representações

```
GET /clientes/1234 HTTP/1.1
Host: xyz.com
Accept: application/meucliente+xml

GET /clientes/1234 HTTP/1.1
Host: xyz.com
Accept: text/x-vcard
```

- Possível uso: uma mesma API que gera as informações em HTML para uma página ou em um formato XML para ser usada por alguma aplicação

REST – Comunicação sem manutenção de estado

- Depois de ver os princípios anteriores, fica fácil entender como fazer isso
- Colocar na URI um ID de sessão não transforma seu web service em um RESTful
- Vantagens
  - Diminuição da carga do servidor
  - Escalabilidade
  - Independência da implementação do servidor
  - Independência entre servidores

HTTP RESTful

- Também tem sido chamado de WebAPI
- Muito usado para fazer mashups
- Um dos pilares para as páginas de conteúdo mais modernas
  - Faz a junção das informações direto no cliente e não mais no servidor
  - Informações sobrepostas e não necessariamente lado a lado

REST

- A especificação propriamente dita não diz nada quanto a HTTP
- Também não especifica o número de operações, apenas determina que a interface deve ser padrão
- Não é demais supor que REST e HTTP tenham se influenciado mutuamente já que o Fielding, criador e descritor do REST também tenha definido muitos dos padrões HTTP

Referências

- [1] ZHANG, Liang-Jie. Services Computing, Pequim, Springer, 2007.
- [2] Web Services Glossary, http://www.w3.org/TR/ws-gloss/
- [3] TILKOV, Stefan. A Brief Introduction to REST, http://www.infoq.com/articles/rest-introduction , 2007
- [4] MARZULLO, Fabio. SOA na Prática, Editora Novatec, 2009