Sistemas Ubíquos

Edição 2019/2020

REST Web Services

Helena Rodrigues

Bibliografia

Distributed Systems, Concepts and Design, George Coulouris, Jean Dollimore and Tim Kindberg, Addison-Wesley, 2012, chapter 9.

Richardson, L. and Ruby, S., 2007, Restful Web Services, O'Reilly.

R. Fielding and R. Taylor, Principled Design of the modern Web architecture, ACM Transactions on Internet Technology, Vol. 2, No 2, May 2002, pp 115-150

REST Web services

Representational State Transfer

Interfaces para serviços Web que aderem aos princípio da arquitetura REST são conhecidos por RESTful

REST Web services

Usam tanto quanto possível os protocolos básicos da *web* e respeita os princípios nos quais estes se baseiam

Todos os serviços RESTful partilham os mesmos métodos standard definidos no HTTP

Qualquer aplicação *web* é um serviço

- Não há necessidade de distinguir entre sites para humanos e sites para consumo programático.
- Apenas as representações podem ser diferentes
- Expor a informação como sendo algo que pode ser usado por outros para objetivos muito diversos (até ser visto num browser)

REST Web services

RPC Web Services

- getUser()
- addUser()
- removeUser()
- updateUser()
- getLocation()
- addLocation()
- removeLocation()
- updateLocation()
- listUsers()
- listLocations()
- findLocation()
- findUser()

Verbos vs Nomes

- *REST*
  - *Recursos*
    - *Location*
    - *Users*
  - *URLs*

http://www.example.org/locations/us/ny/new_york_city

http://www.example.org/users/joao

  - *Métodos standard do HTTP*

    GET, POST, PUT, DELETE, …

http://pt.wikipedia.org/wiki/Representational_State_Transfer

REST Web services

RESTful web services são serviços baseados em protocolos web e assentes em dois princípios fundamentais:

- Informação de seleção é definida no URI
  - Como é que o servidor vai saber qual o recurso sobre o qual a operação deverá ser aplicada?
  - Principio de endereçamento dos recursos
- Informação de método corresponde ao método HTTP e faz parte do pedido HTTP
  - Porque é que o servidor deve enviar este recurso em vez de o apagar?
  - Principio do interface uniforme

Recursos

Qualquer entidade do sistema que mereça/necessite ser referenciada

Algo que pode ter uma representação

Algo sobre o qual um programa possa atuar

Algo que pode ser referido por outros recursos

Tem de ter um URI associado

Representações

Recursos tem de ter uma representação

Representações são estruturas de dados serializadas

XML, JSON, ..

Representações corresponde a informação sobre o estado do recurso

Pode haver várias representações para o mesmo recurso

XML vs JSON, PT vs Eng

Representações podem incluir *links* para outros recursos

Clientes podem seguir os *links* nos recursos como forma de navegação no espaço de informação do servidor (*connectedness*)

Recursos (exemplos)

Exemplos

- Versão 1.0.3 do software
- A última versão do software
- A entrada no weblog referente a 24 de Outubro 2006
- Uma mapa de estradas de Little Rock, Arkansas
- Informação sobre jellyfish
- Uma diretoria de recursos sobre jellyfish
- O próximo número primo a seguir a 1024
- Os próximos 5 números primos após 1024
- As vendas do 4ºtrimestre de 2004
- Uma lista dos bugs ainda não resolvidos na base de dados

REST Web services vs RPC Web services

Qual a informação necessária para desencadear a invocação de um método remoto

- A referência do objeto no qual o método vai ser invocado
- A identificação do método a invocar
- Os parâmetros de invocação

Em REST

- A referência é o URL do recurso
- Os métodos são os métodos standard do HTTP
- Os parâmetros estão embebidos no URL

Recursos (exemplos)

Exemplo de URL para versão 1.0.3 do software

http://www.example.com/software/releases/1.0.3.tar.gz

Escreva um URL para cada um dos seguintes recursos

A última versão do software
A entrada no weblog referente a 24 de Outubro 2006
Uma mapa de estradas de Little Rock, Arkansas
Informação sobre jellyfish
O próximo número primo a seguir a 1024
Os próximos 5 números primos após 1024
As vendas do 4ºtrimestre de 2004
Uma lista dos bugs ainda não resolvidos na base de dados

Recursos (exemplos)

Exemplos de URIs REST

http://www.example.com/software/releases/1.0.3.tar.gz
http://www.example.com/software/releases/latest.tar.g
http://www.example.com/weblog/2006/10/24/0
http://www.example.com/map/roads/USA/AR/Little_Rock
http://www.example.com/wiki/Jellyfish
http://www.example.com/primenumbers/1/1024
http://www.example.com/primenumbers/5/1024
http://www.example.com/sales/2004/Q4
http://www.example.com/bugs/by-state/open

REST Web services

Addressability

- Uma aplicação é *addressable* se expõe partes relevantes dos seus dados como recursos
- A Informação de *scoping* da operação é definida no URL
- Dado que os recursos são expostos através de URIs, a aplicação vai expor um URL por cada parte de informação que pretende expor.
  - Normalmente isto resulta num número de URLs muito largo e impossível de contabilizar

REST Web services

Uniform Interface

HTTP GET

Obter a representação de um recurso

HTTP PUT *to a new* URI, or HTTP POST *to an existing one*

Criar um novo recurso

HTTP PUT *to an existing* URI

Modificar um recurso existente (idempotente):

HTTP DELETE

Apagar um recurso existente

HTTP methods

Resource	GET	PUT	POST	DELETE
Collection URI, such as `http://example.com/resources`	**List** the URIs and perhaps other details of the collection's members.	**Replace** the entire collection with another collection.	**Create** a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.[16]	**Delete** the entire collection.
Element URI, such as `http://example.com/resources/item17`	**Retrieve** a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	**Replace** the addressed member of the collection, or if it doesn't exist, **create** it.	Not generally used. Treat the addressed member as a collection in its own right and **create** a new entry in it.[16]	**Delete** the addressed member of the collection.

http://en.wikipedia.org/wiki/Representational_state_transfer

REST Web services

Statelessness

- Cada pedido HTTP ocorre sem relação com outros pedidos
- Cada pedido tem de incluir toda a informação necessária para que o servidor possa responder
- Caso essa informação seja mesmo necessária, cabe ao cliente enviá-la de novo.
- Maior Robustez, Visibilidade e Escalabilidade
- Maior *overhead* na comunicação

APIs públicos

APIs anunciados publicamente em http://www.programmableweb.com

(não necessariamente grátis)

Tipo de protocolo	APIs
REST	6505
SOAP	2100

Dados disponíveis em 04.11.2013

APIs Web mais populares

API	Description	Category	Mashups
Google Maps	Mapping services	Mapping	2422
Twitter	Microblogging service	Social	752
YouTube	Video sharing and search	Video	652
Flickr	Photo sharing service	Photos	615
Amazon eCommerce	Online retailer	Shopping	416
Facebook	Social networking service	Social	386
Twilio	Telephony service	Telephony	353
Last.fm	Online radio service	Music	226
eBay	eBay Search service	Search	220
Google Search	Search services	Search	184
Microsoft Bing Maps	Mapping services	Mapping	175
Twilio SMS	SMS messaging service	Messaging	172
del.icio.us	Social bookmarking	Bookmarks	160
Yahoo Search	Search services	Search	144
Yahoo Maps	Mapping services	Mapping	135
Google Ajax Search	Web search components	Search	134
Google App Engine	Developer platform	Tools	122
411Sync	SMS, WAP, and email messaging	Messaging	120
Google Homepage	Portal gadgets	Widgets	101
Yahoo Geocoding	Geocoding services	Mapping	99
foursquare	Social networking and city exploration	Social	94
Amazon S3	Online storage services	Storage	87
Google Chart	Chart creation service	Other	84
GeoNames	Geographic name and postal code lookup	Reference	83

http://www.programmableweb.com

Exercício REST Web services

create presence resource

POST http://127.0.0.1:8080/PresenceRESTWS/webresources/presences/193.137.8.01/
Host: 127.0.0.1:8080
Content-Type: text/xml; charset=utf-8
Content-Length: 0

Exercício REST Web services

Create location resource related to a presence resource

POST http://127.0.0.1:8080/PresenceRESTWS/webresources/presences/ips/193.137.8.01/DSI
Host: 127.0.0.1:8080
Content-Type: text/xml; charset=utf-8
Content-Length: 0

Exercício REST Web services

create presence resource (Conceção alternativa)

POST http://127.0.0.1:8080/PresenceRESTWS/webresources/presences/193.137.8.01/
Host: 127.0.0.1:8080
Content-Type: text/xml; charset=utf-8
Content-Length: …

```
<?xml version="1.0" encoding="utf-8"?>
<ipInfo>
<ip>193.137.8.01</ip>
<lastSeen>1416485084775</lastSeen>
<location>DSI</location>
</ipInfo>
```

Exercício REST Web services

Obtain list of presences resources

GET http://127.0.0.1:8080/PresenceRESTWS/webresources/presences HTTP/1.1
Host: 127.0.0.1:8080
Accept: …
User-Agent: …
Accept-Encoding: …
Accept-Language: …

Exercício REST Web services

Tecnologias a utilizar

.NET core (https://dotnet.microsoft.com/download/dotnet-core)
Visual Studio Code (https://code.visualstudio.com/)
Docker Desktop (https://www.docker.com/products/docker-desktop)