Entenda o Representational State T (REST) no Ruby

Desenvolva um cliente RESTful simples com Ruby

M. Jones
Publicado em 31/Ago/2012

A Representational State Transfer (REST) é um estilo de arquitetura para comunicação base clientes conversar com os servidores de maneira única. Em particular, a REST representa re servidor, como Identificador Uniforme de Recursos (URIs), simplificando a implementação d Transport Protocol (HTTP). Começaremos com uma introdução às ideias por trás de REST e representação de dados e implementaremos um cliente REST simples na linguagem Ruby.

Rápida introdução a HTTP

Começaremos com uma rápida introdução a HTTP, uma vez que é importante entendê-lo pa Embora HTTP seja o protocolo de comunicação de base que conecta navegadores da web a para transferir muitos tipos de dados além de HTML.

HTTP é um protocolo de solicitação e resposta—ou seja, os clientes fazem solicitações e os s solicitações com uma resposta. O protocolo real para HTTP é bastante legível para humanos telnet para fazer uma solicitação a um servidor da web.

A Listagem 1 fornece uma solicitação HTTP e uma resposta parcial de um servidor da web. E inicio a solicitação I com o telnet especificando o nome de domínio do servidor da web e a porta (80 é uma porta HTTP típica). O Telnet responde primeiro com uma resolução de

Sistema de Nomes de Domínio do nome de domínio para um endereço IP, e então, indica qu
estou conectado ao servidor da web. Então, especifico uma linha de solicitação (que contém

de solicitação (que podem ser bastante grandes, mas como eu estou digitando, simplesmen
especifico o cabeçalho da solicitação de `Host` , que indica o host e a porta opcional da qual
estou fazendo a solicitação). Minha solicitação é seguida por uma linha em branco, que indic
ao servidor da web que minha solicitação está concluída. O servidor da web então fornece
uma resposta, indicando o protocolo usado e fornecendo um código de status (nesse caso, 2
válida) e cabeçalhos de resposta adicionais. Uma linha em branco então aparece, seguida pc
Essa é uma representação do recurso— nesse caso, um documento HTML.

Listagem 1. Realizando uma transação HTTP com telnet

```
$ telnet mtjones.com 80
Trying 198.145.43.103...
Connected to mtjones.com.
Escape character is '^]'.
GET /index.html HTTP/1.1Host: example.org

HTTP/1.1 200 OK
Date: Sun, 25 Mar 2012 05:33:07 GMT
Server: Apache
Last-Modified: Sat, 26 Sep 2009 20:22:36 GMT
ETag: "2c984bf-798-d3451b00"
Accept-Ranges: bytes
Content-Length: 1944
Vary: Accept-Encoding
Content-Type: text/html

<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" ...
```

Esse exemplo ilustra uma transação simples, mas vários métodos na verdade são implemen
usado para recuperar um recurso do servidor da web, enquanto `HEAD` é usado para obter sor
recurso (não o conteúdo real). Um método `POST` é usado para fornecer novo conteúdo ao ser
usado para colocar dados para um recurso existente no servidor da web. A Tabela 1 fornece
HTTP

Tabela 1. Métodos de solicitação HTTP 1.1 comuns

Método	Idempotente	Descrição
`OPTIONS`	Sim	Solicitação de informações sobre as opções de co
`GET`	Sim	Recupera a representação para um URI

Método	Idempotente	Descrição

PUT	Sim	Cria ou substitui um recurso com uma representac
POST	Não	Aumenta um recurso existente com uma represen
DELETE	Sim	Exclui um recurso que o URI especifica

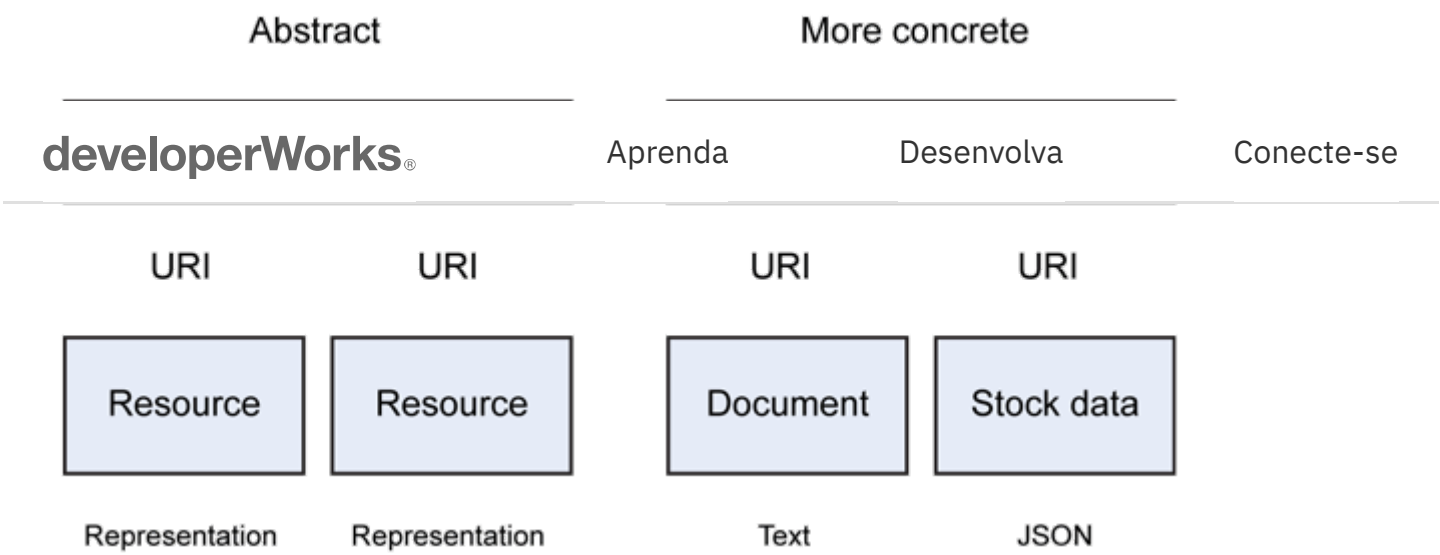
O que é importante observar com essa breve exploração de HTTP é que esse é um protocolo básicas em recursos. Embora HTTP seja comumente usado atualmente para transferir conte clientes, ele também é um protocolo cada vez mais usado para interações de sistema distrib interfaces de programação de aplicativos (APIs) que permitem a sistemas heterogêneos cor

Agora observaremos a pilha de protocolo para explorar a camada REST.

O que é REST?

REST é mais um estilo de arquitetura, e menos uma implementação ou design específico. Uma arquitetura RESTful é definida por um conjunto simples de restrições, ilustradas na Figura 1. . No centro de uma arquitetura RESTful está o conjunto de recursos. Esses recursos são identificados por URIs (como um Localizador Uniforme de Recursos [URL]) e uma representação interna (comumente uma forma de dados autodescritivos, que você explorará em breve). Por fim, há um conjunto de operações através das quais é possível manipular os recursos.

Figura 1. Visualização de alto nível de uma arquitetura RESTful



Em termos mais concretos, esses recursos podem representar objetos de dados usando um JavaScript Object Notation [JSON]). É possível abordar os recursos por meio de URLs (como usando um conjunto de operações padrão (HTTP `GET`, `POST`, `DELETE` e afins). Usar HTTP com desenvolvimento das arquiteturas RESTful, porque elas usam um protocolo de base bem co também está amplamente disponível e não requer nova configuração para usar, incluindo se gateways, proxies, entidades de aplicação de segurança e serviços de armazenamento em c servidores REST altamente escaláveis, é possível explorar outros recursos úteis, como balar

Características de uma arquitetura RESTful

Embora as arquiteturas RESTful tenham liberdade considerável na sua implementação, um importante.

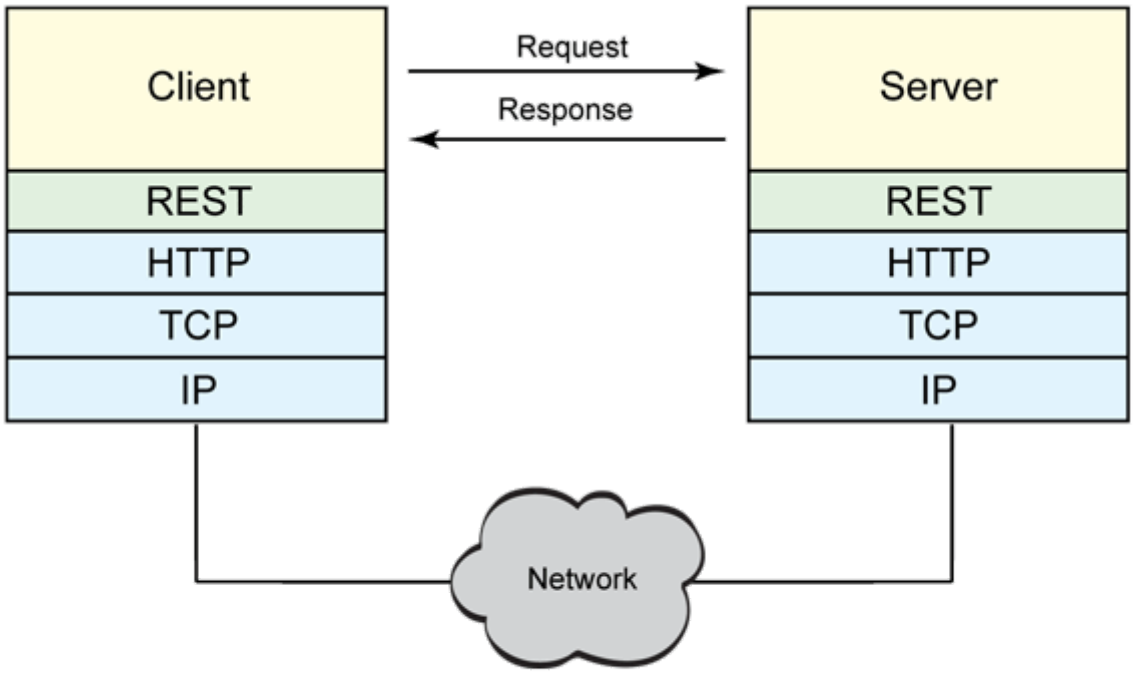
A REST define uma arquitetura de cliente-servidor em que os clientes têm acesso a recursos representação que o servidor exporta. Os clientes não acessam os recursos diretamente, ma representação do recurso através de uma interface uniforme. Como com muitas arquitetura implementada como uma arquitetura em camadas, permitindo a exploração das várias capa inferiores (balanceamento de carga HTTP, etc.) fornecem.

Mas um aspecto essencial das arquiteturas RESTful é que elas são stateless. O servidor não cliente entre transações, e toda transação deve conter todas as informações necessárias pa particular. Essa característica tende a tornar as arquiteturas RESTful mais confiáveis e tamb escalabilidade.

Interface REST de amostra

Observe uma implementação REST de amostra para ilustrar algumas das características da que a REST conta com interações de cliente-servidor (consulte a Figura 2). . Um aplicativo cl

Figura 2. Arquitetura em camadas de interações RESTful



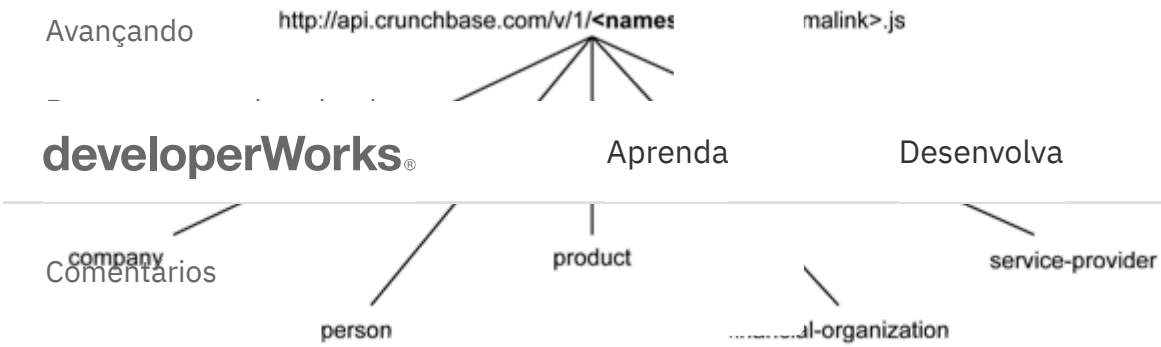
Um exemplo interessante de uma API REST que é possível usar para desenvolver um cliente CrunchBase é um banco de dados gratuito de empresas, pessoas e investidores relacionado um front-end da web tradicional, o CrunchBase oferece uma interface baseada em REST/JS

O CrunchBase implementa três ações através da sua API:

- mostrar (para recuperar informações sobre uma entidade específica)
- procurar (para recuperar uma lista de es que corresponde a um dado critério de
- listar (para recuperar todas as entidac o de um dado namespace)

O CrunchBase também exporta cinco nam s para seus dados —consulte a Figura 3—ju usado para interações REST do CrunchBa amespaces são empresa, pessoa, produto provedor de serviços.) Observe que `/v/1` i ersão da API, que no momento é 1. Obse que indica o nome exclusivo das entidade do banco de dados.

Figura 3. Namespaces na API do CrunchBas



Se você desejasse obter informações atualizadas sobre a IBM, poderia desenvolver uma UR empresa (experimente isto no seu navegador):

```
1 http://api.crunchbase.com/v/1/ibm.js
```

É possível digitar esta URL no navegador, e o navegador renderizará a resposta textual (base (enquanto consome os cabeçalhos HTTP). Observe isso em mais detalhes enquanto explora CrunchBase no formato JSON.

Introdução a dados autodescritivos

Comunicar-se entre sistemas heterogêneos apresenta alguns problemas interessantes, um dados para transferência. As máquinas representam dados de diferentes maneiras (de difer flutuante ao conflito de ordenação de byte padrão). Implementações anteriores incluíam o f (ASN.1) e o protocolo Representação Externa de Dados (XDR) (usado dentro do Sistema de abordagens incluem XML, que codifica dados dentro de documentos formatados em ASCII.

Nos últimos seis anos, o formato JSON aumentou em popularidade. Como o nome implica, J JavaScript e é usado para representar estruturas de dados autodescritivas como matrizes as JSON é um formato de intercâmbio de dados comum e tem suporte em uma variedade de li

Agora veja um exemplo de JSON, em particular— um através da interface REST do CrunchBa interativo (`irb`), que permite experimentar com Ruby em tempo real.

Como mostra a Listagem 2, você começa executando o shell Ruby interativo. Você prepara o módulos (em particular, os componentes JSON e HTTP) e define o seu URI. Observe, aqui, q CrunchBase completa (no namespace de empresa, com um permalink de `ibm`). . Isso é forne `Net::HTTP`, que é um atalho para realizar uma solicitação `GET` no URI especificado (analisad individuais através do método `URI.parse`). Se você emitir `resp.body`, poderá ver os dados J inclui um conjunto de pares de nome-valor (como "nome" e "IBM"). Use o método `JSON.par` uma estrutura de objeto Ruby. Por fim, extraia um valor em particular especificando seu non

Listagem 2. Interagindo com o CrunchBase usando Ruby

```
1  $ irb
```



```
5  => true
6  irb(main):003:0> require 'net/http'
7  => true
8  irb(main):004:0> uri = "http://api.crunchbase.com/v/1/company/ibm.js"
9  => "http://api.crunchbase.com/v/1/company/ibm.js"
10 irb(main):005:0> resp = Net::HTTP.get_response(URI.parse(uri))
11 => #<Net::HTTPOK 200 OK readbody=true>
12 irb(main):006:0> puts resp.body
13 {"name": "IBM",
14  "permalink": "ibm",
15  "crunchbase_url": "http://www.crunchbase.com/company/ibm",
16  "homepage_url": "http://www.ibm.com",
17  "blog_url": "",
18  "blog_feed_url": "",
19  "twitter_username": "",
20  "category_code": "software",
21  "number_of_employees": 388000,
22 ...
23 => nil
24 irb(main):007:0> parsedresp = JSON.parse(resp.body)
25 => {"updated_at"=>"Wed Feb 01 03:10:14 UTC 2012", "alias_list"=>nil,
26 ...
27 irb(main):008:0>
28 irb(main):009:0* puts parsedresp['founded_year']
29 1896
30 => nil
31 irb(main):010:0>
```

Na Listagem 2, é possível ver como foi fácil fazer o protótipo de uma extração rápida de dad linhas). Leve isso além agora e desenvolva uma API simples e reutilizável para interagir com

Desenvolvendo um cliente REST simples

Antes de desenvolver seu cliente REST, você deve instalar algumas coisas. Se não tiver Ruby o Ubuntu, emprego a Ferramenta do Pacote Avançado para a maioria desses requisitos de ir pacotes de gem Ruby para outros).

Pegue o pacote Ruby com:

```
1  $ sudo apt-get install ruby
```

Opcionalmente, peque o Shell Ruby Interativo `irb`), que é uma maneira útil de experimentar

```
1  $ sudo apt-get install irb
```

Por fim, a gem JSON é necessária para Ruby. O código a seguir mostra como obter tanto o fr


```
3 $ sudo gem install json
```

Com seu ambiente pronto, comece a desenvolver a API do cliente REST com Ruby. Você viu com um servidor HTTP em Ruby e como analisar um nome simples de um objeto JSON. Des conhecimento um conjunto de classes Ruby que implementam uma API simples.

Primeiro, vejas as duas classes de amostra que interagem com o servidor REST do CrunchBa namespace de empresa e a segunda foca no namespace de pessoa. Observe que ambas est métodos, mas são facilmente extensíveis para os outros elementos de dados.

A Listagem 3 fornece a API para interagir com o namespace de empresa. Essa classe estend (`initialize`) e quatro métodos usados para extrair dados de um registro de empresa basea para a classe é que o usuário cria uma instância do objeto, fornecendo um nome de empres construtor, o CrunchBase é solicitado para o registro da empresa. Você desenvolve dinamica nome da empresa enviado como parte do método `new` . O método `get_response` é usado pa resultado analisado (um objeto hash) é carregado para uma variável da instância (`@record`).

Com o registro analisado disponível, cada método, quando chamado, simplesmente extrai o para o usuário. Os métodos `founded_year`, `num_employees` e `company_type` são diretos, dad método `people` requer um pouco mais de interpretação.

A resposta JSON do CrunchBase é representada por um hash contendo alguns números de primeiros métodos, você especifica a chave para retornar o valor. O método `people` itera por `relationships`. . Cada registro contém uma chave `is_past` (que é usada para identificar se determinada empresa), uma chave `title` e uma chave person, que contém `first_name` , la simplesmente passa por cada uma, e quando a chave `is_past` é falsa, extrai a pessoa e o ca dessas informações. Esse hash é retornado quando mais nenhuma chave é encontrada. Obs esse hash dentro do IRB simplesmente emitindo a resposta com o item `JSON.parse`.

Listagem 3. API CrunchBase de Ruby para o namespace da empresa (company.rb)

```
1 require 'rubygems'
2 require 'json'
3 require 'net/http'
4
5 class Crunchbase_Company
6
7   @record = nil
8
```

```
  def initialize( company )

    base_url = "http://api.crunchbase.com"
```



```
    @record = JSON.parse(resp.body)

      enddef founded_year
    return @record['founded_year']
  end

  def num_employees
    return @record['number_of_employees']
  end

  def company_type
    return @record['category_code']
  end

  def people

    employees = Hash.new

    relationships = @record['relationships']

    if !relationships.nil?

        relationships.each do | person |
            if person['is_past'] == false then
                permalink = person['person']['permalink']
                title = person['title']
                employees[permalink] = title
            end
        end

    end    return employees

  endend
```

A Listagem 4 fornece uma função similar à classe `Company` discutida na Listagem 3. O constr
maneira, e você tem dois métodos simples para extrair o nome da pessoa determinada (do p
método `companies` itera o hash para a chave `relationships` e retorna as firmas (empresas)
estava associada no passado. As empresas associadas são retornadas nesse caso como um
permalinks).

Listagem 4. API CrunchBase de Ruby para o namespace da pessoa (person.rb)

```
require 'rubygems'
require 'json'
require 'net/http'

class Crunchbase_Person

  @record = nil

  def initialize( person )
```

```
    base_url = "http://api.crunchbase.com"
    url = "#{base_url}/v/1/person/#{person}.js"

```



```
  enddef fname
    return @record['first_name']
  end

  def lname
    return @record['last_name']
  end

  def companies

    firms = Array.new

    @record['relationships'].each do | firm |

        firms << firm['firm']['permalink']

end    return firms

  endend
```

Observe que nessas duas classes simples, a Ruby oculta a complexidade de lidar com o serv complexidade de analisar a resposta JSON é totalmente simplificada por meio da gem JSON aplicativos dessas APIs para ilustrar seu uso.

Desenvolvendo alguns aplicativos simples

Inicie com uma demonstração das classes. No primeiro exemplo (consulte a Listagem 5), vo associadas a uma dada empresa (com base no permalink da empresa). O registro recuperad contém uma lista de permalinks para pessoas associadas à empresa. Você itera esse hash d do indivíduo baseado no permalink. Esse registro fornece o primeiro e o último nome do indi registro da empresa (retornado como parte do hash nome-cargo).

Listagem 5. Identificando pessoas associadas a uma empresa (people.rb)

```
#!/usr/bin/ruby

load "company.rb"
load "person.rb"

# Get argument (company permalink)
input = ARGV[0]

company = Crunchbase_Company.new(input)

people = company.people

```

```
# Iterate the people hash
people.each do |name, title|

```



```
    # Emit the name and title
    print "#{person.fname} #{person.lname} | #{title}\n"

endpeople = nil
company = nil
```

O script é executado na Listagem 5 , como mostra a a Listagem 6. . Com o script, você fornec resultado é o nome e sobrenome do indivíduo mais o cargo.

Listagem 6. Testando o script people.rb

```
$ ./people.rb emulex
Jim McCluney | President and CEO
Michael J. Rockenbach | Executive Vice President and CFO
Jeff Benck | Executive Vice President & COO
$
```

Agora analisaremos um exemplo mais complexo. Esse exemplo pega uma determinada emp executiva. Então, identifica as empresas em que esses executivos trabalharam no passado. chamado *influence.rb.* . Como mostrado, você aceita um nome da empresa como o argumen e depois recupera um hash das pessoas atualmente nessa empresa (por meio do método pe e identifica os chefes pelo cargo (não de forma totalmente precisa, dada a variabilidade de c quaisquer chefes identificados, você emite as empresas para as quais esses indivíduos trab empresas do registro da pessoa).

Listagem 7. Relacionamento e influência executiva (influence.rb)

```
#!/usr/bin/ruby

load "crunchbase.rb"

input = ARGV[0]

puts "Executive Relationships to " + input

company = Crunchbase_Company.new(input)

people = company.people

# Iterate through everyone associated with this company
people.each do |name, title|

    # Search for only certain titles
    if title.upcase.include?("CEO") or
       title.upcase.include?("COO") or
       title.upcase.include?("CFO") or
       title.upcase.include?("CHIEF") or
```

```
        title.upcase.include?("CTO") then

         person = Crunchbase_Person.new( name )
```



```
         companies.each do | firm |
             if input != firm
                 puts "   " + firm
             end
         end

endend
```

A Listagem 8 ilustra esse script para a empresa de big data, Cloudera. Como é possível ver n ocultam muitos detalhes de você, permitindo-lhe focar na tarefa em questão.

Listagem 8. Testando o script Ruby de influência

```
$ ./influence.rb cloudera
Executive Relationships to cloudera
  accel-partners
  bittorrent
  mochimedia
  yume
  lookout
  scalextreme
  vivasmart
  yahoo
  facebook
  rock-health
$
```

Usando outros métodos HTTP REST

Nos exemplos simples mostrados aqui, você usou somente o método `GET` para extrair dados Outros sites podem estender uma interface para recuperar e enviar dados para o servidor RI fornece uma introdução a outros métodos `Net::HTTP` .

Listagem 9. Métodos HTTP para outras interações RESTful

```
http = Net::HTTP.new("site url")

# Delete a resource
transaction = Net::HTTP::Delete.new("resource")
response = http.request(transaction)

# Post a resource
resp, data = Net::HTTP.post_form( url, post_arguments )

# Put a resource
transaction = Net::HTTP::Put.new("resource")
transaction.set_form_data( "form data..." )
```

```
13 response = http.request(transaction)
```


(consulte Recursos). Essa gem fornece uma camada REST sobre HTTP, oferecendo metodos

Avançando

Espero que essa rápida introdução aos princípios de REST com Ruby ilustre a potência da ar como por que Ruby é uma das minhas linguagens favoritas. A REST é uma das arquiteturas c nuvens de computação e armazenamento e, portanto, vale a pena dedicar tempo para enter Recursos, você encontrará links para mais informações sobre as tecnologias usadas neste a artigos sobre REST no developerWorks.

Recursos para download

PDF desse conteúdo

Temas relacionados

- REST é uma arquitetura de software para comunicação entre sistemas distribuídos. A RE vez por Roy Fielding em sua dissertação de doutorado, chamada Architectural Styles and Software Architectures, na Universidade da California, Irvine.
- REST é desenvolvida sobre HTTP. Saiba mais detalhes sobre HTTP na Internet Engineeri Comentários-2616 (para a versão 1.1).
- Obtenha uma introdução interessante a REST no vídeo Intro to REST, fornecido por Joe (
- Saiba mais sobre várias ideias e tecnologias úteis exploradas neste artigo: URI, URL e JS
- Ruby é de longe minha linguagem de script favorita e uma das linguagens mais intuitivas apenas para a criação de protótipos rápida e desenvolvimento experimental, mas também software de produção. Se for iniciante em Ruby, você irá se agradecer mais tarde por rea Ruby, Ruby in Twenty Minutes.
- Leia mais sobre REST nestes artigos do developerWorks:
  - RESTful Web services: The basics (Alex Rodriguez, novembro de 2008) fornece uma propriedades básicas.

- Programação de aplicativo REST (Paul Sonnenberg, setembro de 2010) foca em dese
em REST.

- Para uma implementação ainda mais simples de REST em Ruby, experimente a gem `res` cliente REST simplifica a maioria das grandes operações e permite que você foque no se
- Avalie os produtos IBM da maneira que for melhor para você: faça download da versão d produto online, use-o em um ambiente de nuvem ou passe algumas horas na SOA Sandb implementar arquitetura orientada a serviço (SOA) de maneira eficiente.

Comentários

Acesse ou registre-se para adicionar e acompanhar os comentários.

Receba notificações dos comentários