UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO - MESTRADO

REST - Representational State Transfer

Disciplina: Programação Paralela e Distribuída

Professor: Sérgio Teixeira de Carvalho

Aluno: Divino Alves Ferreira Júnior

Abril - 2019

Roteiro

- Conceito
- Histórico
- Fundamentação do modelo
- Elementos arquiteturais
- Descomplicando

CONCEITO

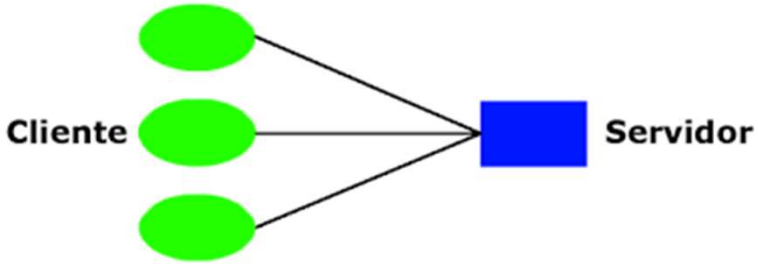
- REST - Representational State Transfer (Estado de transferência representacional)
- Independe de tecnologia
- **Não** é API, FRAMEWORK – é um modelo arquitetural
- Usado em Web Services
- Base HTTP

HISTÓRICO

- Apresentado em 2000 em uma tese de doutorado
- Por Roy Fieldeing
  - é um cientista da computação americano, um dos principais autores da especificação HTTP. (...) Ele é uma autoridade em arquitetura de rede de computadores e co-fundador do projeto Apache HTTP Server (Wikipédia - https://en.wikipedia.org/wiki/Roy_Fielding - acessado em abril/2019)

REST – Porque faz tanto sentido hoje?

- Necessidade de integrar – cada vez mais empresas contratam softwares diferentes;
- Dispositivos diferentes acessando nossas aplicações: Smartphones, Smarttv, IoT;
- Sistemas distribuidos



REST - Fundamentação

- REST (Representation State Transfer) é um estilo arquitetural híbrido para sistemas distribuídos derivado da combinação de alguns estilos arquiteturais com algumas características adicionais. [FIELDING, 2000].
- Partiu de um ponto *Null Style* – representa um conjunto simples e vazio de característica

REST - Fundamentação

- Primeira característica (*Null Style)* - estilo arquitetural cliente – servidor
  - Princípio é a separação de responsabilidades
  - Permite que os componentes se desenvolvam de forma independente
- Segunda característica - Stateless – sem estado
  - A comunicação não depende de estados controlados pelo servidor. Somente o cliente deve conhecer esta informação – **maior escalabilidade e performance**.

REST - Fundamentação

- Terceira característica – Cache – combinando as características temos : *Cliente com cache - Servidor sem estado*
  - Pode eliminar parcialmente ou totalmente algumas interações com o servidor;
  - Cliente pode ou não usar esta informação do cache.

REST - Fundamentação

- Quarta característica – Interface uniforme entre componentes
  - Toda troca de informação é feita de forma genérica
  - 4 elementos de interface - Identificação de Recursos; Manipulação de Recursos através de Representações; Mensagens Auto-descritivas e Hipermídia como Máquina de Estados da Aplicação

REST - Fundamentação

- Quinta característica – separação em camadas
  - Composta por diversos níveis hierárquicos;
  - O conhecimento que o sistema possui é restrito a sua camada somente;
  - Provê maior simplicidade e flexibilidade na comunicação.
- Sexta característica – código sob demanda – opcional
  - Funcionalidade inseridas no tempo de execução;
  - Objetivo de simplificar e ampliar a capacidade dos clientes

REST – Elementos Arquiteturais

- Elementos de Dados
  - A natureza e o estado dos elementos de dados são aspectos chave
  - Identificador global– Ex. *URI* http://servicorest.com.br/produtos;
  - Recurso – abstração sobre determinado tipo de informação

Data Element	Modern Web Examples
resource	the intended conceptual target of a hypertext reference
resource identifier	URL, URN
representation	HTML document, JPEG image
representation metadata	media type, last-modified time
resource metadata	source link, alternates, vary
control data	if-modified-since, cache-control

Fonte: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

REST – Elementos Arquiteturais

- Conectores
  - Responsável por transmitir o pedido (cliente, servidor, cache)

Connector	Modern Web Examples
client	libwww, libwww-perl
server	libwww, Apache API, NSAPI
cache	browser cache, Akamai cache network
resolver	bind (DNS lookup library)
tunnel	SOCKS, SSL after HTTP CONNECT

Fonte: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

REST – Elementos Arquiteturais

- Componentes
  - Clientes e servidores – comunicam por interface HTTP trocando representação de recursos (arquivos ou ficheiros recebidos e enviados)

Component	Modern Web Examples
origin server	Apache httpd, Microsoft IIS
gateway	Squid, CGI, Reverse Proxy
proxy	CERN Proxy, Netscape Proxy, Gauntlet
user agent	Netscape Navigator, Lynx, MOMspider

Fonte: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

REST - Descomplicando

- Aplicação pode interagir com um recurso conhecendo o identificador do recurso e a ação requerida, não necessitando conhecer se existem caches, proxys, ou outra, entre ela e o servidor que guarda a informação.
- REST – modelo arquitetural
- RESTFul – denominação dada a uma aplicação baseada no modelo arquitetural REST

REST - Descomplicando

- frequentemente aplicado à web services - APIs
- Usa protocolo HTTP e seus métodos (GET, POST, PUT e DELETE)
- URIs são usados para expor a estrutura do serviço.
- Trabalha essencialmente com componentes, conectores e dados.
- Leveza dos pacotes de dados transmitidos na rede - encapsulamento JSON, XML, YAML ...

Método-CRUD	Método-HTTP
Create	POST
Read	GET
Update	PUT
Delete	DELETE

REST - Descomplicando

- Sem estado
  - Cada comunicação é independente
  - Tudo é passado e recebido de uma vez (request/response)
  - Neste caso o cliente sempre deve enviar seu token ao servidor
- Deve facilitar o cache de conteúdo no cliente.
- Definição clara do papel cliente e do servidor – se torna mais escalável
- Uso em camadas facilita: escalabilidade, confiabilidade e segurança.

REST – Invocação e resposta de um método pela visão tradicional de Serviços Web

```
<?xml version="1.0" ?>
<soap:Envelope xmlns:soap=" http://www.w3.org/2003/05/soap-envelope">
    <soap:Body>
        <u:getStudentInfo xmlns:u="http://univ.example.com">
            <u:code>123456789</u:code>
        </u:getStudentInfo>
    </soap:Body>
</soap:Envelope>
```

```
<?xml version="1.0" ?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
    <soap:Body>
        <u:getStudentInfoResponse xmlns:u="http://univ.example.com">
            <u:code>123456789</u:code>
            <u:name>João Silva</u:name>
            <u:age>19</u:age>
        </u:getStudentInfoResponse>
    </soap:Body>
</soap:Envelope>
```

Fonte: http://www.monografias.nrc.ice.ufjf.br/tcc-web/exibePdf?id=17

REST – Invocação e resposta de um método pela visão REST de Serviços Web

```
GET http://univ.example.com/studentInfo/123456789
```

```
<?xml version="1.0" ?>
<u:StudentInfo xmlns:u="http://univ.example.com">
    <u:code>123456789</u:code>
    <u:name>João Silva</u:name>
    <u:age>19</u:age>
</u:StudentInfo>
```

Fonte: http://www.monografias.nrc.ice.ufjf.br/tcc-web/exibePdf?id=17

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO - MESTRADO

REST - Representational State Transfer

Disciplina: Programação Paralela e Distribuída

Professor: Sérgio Teixeira de Carvalho

Aluno: Divino Alves Ferreira Júnior

Abril - 2019