Programación Distribuida

Servicios REST

JAIME SALVADOR

AGENDA

- **Introducción**
- Recursos
- Métodos HTTP
- HTTP Status Codes

Introducción

- REST

  Representational State Transfer (REST)

RPC: orientado a servicios (acciones y verbos)

REST: orientado a recursos (cosas y nombres)

Introducción

- Representational

  Los recursos REST pueden ser representados de cualquier forma, incluyendo XML, JSON (JavaScript Object Notation), HTML

- State

  Interesa conocer el estado de un recurso (no interesa las acciones que se pueden realizar sobre un recurso)

- Transfer

  Transferencia los datos asociados a un recurso, en algún tipo de representación, de una aplicación a otra

Introducción

REST trata sobre la **transferencia del estado de un recurso** (en cualquier forma, siempre la más adecuada) desde un servidor hacia un cliente.

REST es una arquitectura para el diseño de aplicaciones distribuidas.

- **REST:** arquitectura que se ejecuta sobre HTTP
- **RESTful:** servicio web que implementa la arquitectura REST

Introducción

Principios REST:

- **Cliente/servidor**: Separación entre cliente y servidor, de tal forma que los dos pueden evolucionar de forma independiente
- **Stateless** (sin estado): La comunicación entre cliente y servidor debe ser sin mantener un estado previo.
- **Sistema basado en capas**: Múltiples capas jerárquicas (gateways, firewalls, proxies) pueden existir entre cliente y servidor. Las capas pueden ser agregadas/eliminadas transparentemente

Introducción

Principios REST:

- **Cache**: La información enviada desde el servidor puede ser o no almacenada en un caché
- **Interface Uniforme**: comunicación a través de interfaces (recursos, representación, URIs, métodos HTTP)

  HATEOAS: Hypermedia as the engine of application state
- **Code on demand**: el cliente puede descargar bajo demanda código desde el servidor y ejecutarlo (OPCIONAL)

Introducción

Una aplicación que cumple con los seis principios anteriores es considerada una aplicación RESTful

AGENDA

- Introducción
- **Recursos**
- Métodos HTTP
- HTTP Status Codes

Recursos

Recursos

Identificadores

Metadatos

Representación

Metadatos de la representación

Datos de controls

Recursos

Recursos

- Un recurso es cualquier cosa que puede ser accedido o manipulado: videos, imágenes, perfiles de usuarios
- Los recursos pueden depender de otros recursos

  ejemplo: cliente - pedido (el pedido pertenece al cliente)
- Pueden ser agrupadas en colecciones

  Ejemplo: los pedidos de un cliente

Recursos

Identificadores

- Uniform Resource Identifier (URI)

  scheme:scheme-specific-part

URI	Resource Description
http://blog.example.com/posts	Represents a collection of blog post resources
http://blog.example.com/posts/1	Represents a blog post resource with identifier "1"; such resources are called singleton resources
http://blog.example.com/posts/1/comments	Represents a collection of comments associated with the blog entry identified by "1"; collections such as these that reside under a resource are referred to as subcollections
http://blog.example.com/posts/1/comments/245	Represents the comment resource identified by "245"

Recursos

Identificación

- URI template

`http://blog.example.com/{year}/posts`

{} representa la parte variable

Recursos

Representación

- Los recursos REST son entidades abstractas
- Los datos y metadatos que comprenden un recurso REST deben ser serializados en una representación que el cliente pueda entender

  HTML, XML, JSON
- Ningún cliente interactúa con un recurso, interactúa con su REPRESENTACION

Recursos

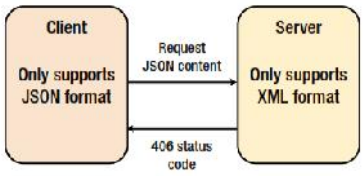
Representación

```
<Customer>
   <id>123</id>
   <name>John</name>
</Customer>
```

```
{
   "Customer":
   {
      "id":"123",
      "name":"John“
   }
}
```

Recursos

Representación



Recursos

Data Element	Description
Resource	Conceptual target of a hypertext reference, e.g., customer/order
Resource Identifier	A uniform resource locator (URL) or uniform resource name (URN) identifying a specific resource, e.g., http://myrest.com/customer/3435
Resource Metadata	Information describing the resource, e.g., tag, author, source link, alternate location, alias names
Representation	The resource content—JSON Message, HTML Document, JPEG Image
Representation Metadata	Information describing how to process the representation, e.g., media type, last-modified time
Control Data	Information describing how to optimize response processing, e.g., if-modified-since, cache-control-expiry

AGENDA

- Introducción
- Recursos
- **Métodos HTTP**
- HTTP Status Codes

Métodos HTTP

- El principio "Interface Uniforme" restringe la interacción entre el cliente y servidor a través de operaciones estandarizadas o **verbos**
- El estándar HTTP proporciona métodos HTTP los cuales permiten a los clientes interactuar y manipular recursos: GET, POST, PUT, DELETE

Métodos HTTP

Safety

- Un método HTTP se dice que es "seguro" si no causa ningún cambio en el "estado" del servidor.
  - GET o HEAD: se utilizan para recuperar información/recursos desde el servidor
  - Son operaciones de solo-lectura
- El método NO necesariamente devuelve la misma información (representación) cada vez que se invoca

Métodos HTTP

Idempotency

- Una operación se considera idempotente si produce la misma salida sin importar el número de veces que se ejecute el método
  - GET, HEAD, PUT, DELETE se consideran idempotentes
  - Garantizan que el cliente pueda repetir la invocación del método y esperar el mismo efecto que se generaría si solo se invoca una sola vez

POST no se considera un método idempotente

Métodos HTTP

GET

- Se utiliza para recuperar la representación de un recurso

```
http://blog.example.com/posts/1
http://blog.example.com/posts
```

GET: safe, idempotent

Métodos HTTP

GET

```
GET      /posts/1 HTTP/1.1
Accept:  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Host: blog.example.com
```

Métodos HTTP

GET

```
GET      /posts/1 HTTP/1.1
Accept:  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Host: blog.example.com
```

```
Content-Type: text/html; charset=UTF-8
Date: Sat, 10 Jan 2015 20:16:58 GMT
Server: Apache
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <title>First Post</title>
      </head>
      <body>
           <h3>Hello World!!</h3>
      </body>
</html>
```

Métodos HTTP

DELETE

- Se utiliza para solicitar la eliminación de un recurso
- Si a continuación se intenta recupera un recurso mediante GET, el servidor retorna "Resource Not Found" (status 404)

DELETE: idempotent

Métodos HTTP

DELETE

```
Delete /posts/1  HTTP/1.1
Content-Length: 0
Content-Type: application/json
Host: blog.example.com
```

```
Delete /posts/2/comments HTTP/1.1
Content-Length: 0
Content-Type: application/json
Host: blog.example.com
```

Métodos HTTP

PUT

- Se utiliza para modificar el estado de un recurso
- El cliente envía una versión modificada del recurso al servidor. El servidor reemplaza el estado del recurso con la nueva información

PUT: idempotent

Métodos HTTP

PUT

```
PUT /posts/1     HTTP/1.1

Accept: */*
Content-Type: application/json
Content-Length: 65
Host: blog.example.com

BODY

{"title": "First Post","body": "Updated Hello World!!"}
```

Métodos HTTP

POST

- Se utiliza para crear un recurso
- No necesita conocer el URIs del recurso (id)

POST:

Métodos HTTP

POST

```
POST /posts      HTTP/1.1

Accept: */*
Content-Type: application/json
Content-Length: 63
Host: blog.example.com

BODY

{"title": "Second Post","body": "Another Blog Post."}
```

```
Content-Type: application/json
Location: posts/12345
Server: Apache
```

Métodos HTTP

CRUD AND HTTP VERBS

Data-driven applications typically use the term CRUD to indicate four basic persistence functions—Create, Read, Update, and Delete. Some developers building REST applications have mistakenly associated the four popular HTTP verbs GET, POST, PUT, and DELETE with CRUD semantics. The typical association often seen is:

```
Create -> POST
Update -> PUT
Read -> GET
Delete -> DELETE
```

These correlations are true for Read and Delete operations. However, it is not as straightforward for Create/Update and POST/PUT. As you have seen earlier in this chapter, PUT can be used to create a resource as long as idempotency constraint is met. In the same way it was never considered non-RESTful if POST is used for update (http://roy.gbiv.com/untangled/2009/it-is-okay-to-use-post). It is also possible for a client to use PATCH for updating a resource.

Therefore, it is important for API designers to use the right verbs for a given operation than simply using a 1-1 mapping with CRUD.

AGENDA

- Introducción
- Recursos
- Métodos HTTP
- **HTTP Status Codes**

HTTP Status Codes

Permiten al servidor comunicar al cliente el resultado del proceso de la petición realizada

- Informational Codes: serie 100

  El servidor ha recibido la petición pero no ha terminado de procesarla

- Success Codes: serie 200

  La petición ha sido recibida y procesada con éxito

HTTP Status Codes

- Redirection Codes: serie 300

  La petición ha sido procesada, pero el cliente debe realizar una acción adicional para completar la petición

- Cliente Error Codes: serie 400

  Indican que existe un error en la petición del cliente

- Server Error Codes: serie 500

  Indican que existe un error en el servidor mientras se procesaba la petición

HTTP Status Codes

Status Code	Description
100 (Continue)	Indicates that the server has received the first part of the request and the rest of the request should be sent.
200 (OK)	Indicates that all went well with the request.
201 (Created)	Indicates that request was completed and a new resource got created.
202 (Accepted)	Indicates that request has been accepted but is still being processed.
204 (No Content)	Indicates that the server has completed the request and has no entity body to send to the client.
301 (Moved Permanently)	Indicates that the requested resource has been moved to a new location and a new URI needs to be used to access the resource.
400 (Bad Request)	Indicates that the request is malformed and the server is not able to understand the request.
401 (Unauthorized)	Indicates that the client needs to authenticate before accessing the resource. If the request already contains client's credentials, then a 401 indicates invalid credentials (e.g., bad password).

HTTP Status Codes

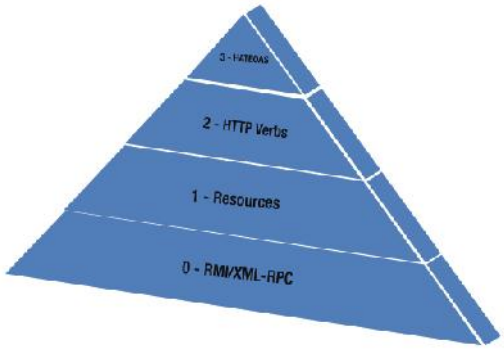
403 (Forbidden)	Indicates that the server understood the request but is refusing to fulfill it. This could be because the resource is being accessed from a blacklisted IP address or outside the approved time window.
404 (Not Found)	Indicates that the resource at the requested URI doesn't exist.
406 (Not Acceptable)	Indicates that the server is capable of processing the request; however, the generated response may not be acceptable to the client. This happens when the client becomes too picky with its accept headers.
500 (Internal Server Error)	Indicates that there was an error on the server while processing the request and that the request can't be completed.
503 (Service Unavailable)	Indicates that the request can't be completed, as the server is overloaded or going through scheduled maintenance.

AGENDA

Richardson's Maturity Model

Clasifica los servicios web basados en REST en qué tanto se adhieren a los 6 principios REST

HTTP Status Codes



HTTP Status Codes

Level Zero

- Este es el nivel de madurez más rudimentario para un servicio
- Los servicios utilizan HTTP como protocolo de transporte y realizan RPC sobre un único URI
- Solamente utilizan GET o POST
- Ejemplo: SOAP

HTTP Status Codes

Level One

- Introduce múltiples URIs, uno por recurso
- Solamente utilizan POST

HTTP Status Codes

Level Two

- Los servicios en este nivel utilizan los métodos HTTP de forma correcta
- Los servicios utilizan los códigos de estados HTTP

Ejemplo: implementación de CRUD

HTTP Status Codes

Level Three

- Este es el nivel más maduro de un servicio web
- El servicio web está construido bajo la noción de HATEOAS (Hypermedia as the Engine of Application State)
- Los servicios web proporcionan información adicional (links) relacionados con otros recursos e informan al cliente qué acción tomar a continuación

PREGUNTAS?