RESTFul Web Services
(An Enabler of Cloud Computing)
Fundamentals

Roch Glitho, PhD
Full Professor
Ericsson / ENCQOR-5G Senior Industrial Research Chair
Cloud and Edge Computing for 5G and Beyond
My URL - http://users.encs.concordia.ca/~glitho/
Outline

1. Web services in general
2. Detailed presentation of REST
3. REST Case studies
   - Hypervisor/Containers API
   - Openstack compute API
   - Messaging
   - Conferencing
Web Services in general
Web Services in General

1. Definition and principles

2. Web services and Cloud Computing

2. Technologies
Web Services

- RESTful Web Services (Focus of this course)
- SOAP – BASED WEB SERVICES (Legacy – less and less used)
Definitions and principles

“The term Web Services refers to an architecture that allows applications (on the Web) to talk to each other. Period. End of statement”

Adam Bobsworth in ACM Queue, Vol1, No1
Definitions and principles

The three fundamental principles, still according to Adam Bobsworth:

1. Coarse grained approach (i.e. high level interface)
2. Loose coupling (e.g. application A which talks to application B should not necessarily be re-written if application B is modified)
3. Synchronous mode of communication, but also asynchronous mode
Web service and Cloud: Illustration
IaaS level (Openstack REST interface)

1: GET:/servers

2:

```
{ "servers": [ { "id": "1", "links": [ { "href": "http://openstack.example.com/v2/6f70656e737461636b20342065766572/servers/1", "rel": "self" }, … } ] }
```

3: POST:/servers

2:

```
{ "server": { "name": "auto-allocate-network", "imageRef": "70a599e0-31e7-49b7-b260-868f441e862b", "flavorRef": "http://openstack.example.com/flavors/1", "networks": "auto" } }
```

Create Server
Web services and Cloud Computing
(Illustration at the SaaS level)
Zoom Developer API
https://developers.zoom.us/docs/api/ (accessed on September 27, 2023)

Meeting
Overview
REST API
Webhooks
Master account API

Zoom Phone
Overview
REST API
Webhooks
Master account API
Web services and Cloud Computing
(Illustration at the SaaS level)
Zoom Developer blog (Processing Zoom audio in real time in order to feed it to an AI box such as NLP box)
https://developers.zoom.us/blog/windows-msdk-realtimetime audio/ (accessed on September 27, 2023)

Underlying architecture
Demos
Code
Technologies

Protocol:
HTTP (Both REST and SOAP Based Web services)

Data format
- XML (Both REST and SOAP)
- Other formats (REST only) – Some examples:
  - JSON
  - Plain Text
HTTP

HTTP (HyperText Transfer Protocol)
- Is an application-level protocol for distributed, collaborative, hypermedia information systems
  - HTTP has been in use since 1990
  - HTTP is a request-response protocol
  - HTTP requests relates to resources
    - A resource is any object or service network that can be identified by a URI (Universal Resource Identifier)
HTTP

Client
- A program that establishes connections for the purpose of sending requests

User Agent
- The client which initiates a request (e.g. browser)

- Note
  - A request may pass through several servers
HTTP

Server

- An application program that accepts connections in order to service requests by sending back responses
- A given program may be capable of being both a client and a server
- The role depends on connections
HTTP

- **Origin server**
  - The server on which a given resource resides or is to be created

- **Proxy server**
  - An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients

- **Gateway server**
  - receives requests as if it were the origin server for the requested resource, and forwards the request to another server
  - Is transparent to the client
HTTP

HTTP-message = Request | Response

generic-message = start-line
*(message-header CRLF)
CRLF
[ message-body ]

start-line = Request-Line | Status-Line
HTTP

HEAD
- retrieve meta-information about a web page, without retrieving the page content (ex: get the date for last modification)

GET
- retrieve the page content

PUT
- store the enclosed content under the supplied Request-URI

POST
- add the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI
  - E.g.
    - Post a message to a mailinglist
    - Extend a database by appending information
    - Transfer a form data
HTTP

DELETE
- Deletes the page

TRACE
- Debug

OPTIONS
- Allows the client to discover the options supported by the server

CONNECT
- Not used currently
HTTP

The built-in HTTP request methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Request to read a Web page</td>
</tr>
<tr>
<td>HEAD</td>
<td>Request to read a Web page’s header</td>
</tr>
<tr>
<td>PUT</td>
<td>Request to store a Web page</td>
</tr>
<tr>
<td>POST</td>
<td>Append to a named resource (e.g., a Web page)</td>
</tr>
<tr>
<td>DELETE</td>
<td>Remove the Web page</td>
</tr>
<tr>
<td>TRACE</td>
<td>Echo the incoming request</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Reserved for future use</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>Query certain options</td>
</tr>
</tbody>
</table>
The status code response groups.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xx</td>
<td>Information</td>
<td>100 = server agrees to handle client’s request</td>
</tr>
<tr>
<td>2xx</td>
<td>Success</td>
<td>200 = request succeeded; 204 = no content present</td>
</tr>
<tr>
<td>3xx</td>
<td>Redirection</td>
<td>301 = page moved; 304 = cached page still valid</td>
</tr>
<tr>
<td>4xx</td>
<td>Client error</td>
<td>403 = forbidden page; 404 = page not found</td>
</tr>
<tr>
<td>5xx</td>
<td>Server error</td>
<td>500 = internal server error; 503 = try again later</td>
</tr>
</tbody>
</table>
Detailed presentation of REST
RESTful Web Services

1. Introduction
2. Resource Oriented Architecture
3. Resources
4. Properties
5. Tool kits
6. Examples of RESTful Web services
Introduction

- What about using the Web’s basic technologies (e.g. HTTP) as a platform for distributed services?
  - This is what is REST about.
Introduction

- REST was first coined by Roy Fielding in his Ph.D. dissertation in 2000

- It is a network architectural style for distributed hypermedia systems.
Introduction

- REST is a way to reunite the programmable web with the human web.

- It is simple
  - Uses existing web standards
  - The necessary infrastructure has already become pervasive
  - RESTful web services are lightweight
  - HTTP traverse firewall
Introduction

- RESTful web services are easy for clients to use
- Relies on HTTP and inherits its advantages, mainly
  - Statelessness
  - Addressability
  - Unified interface
Resource-Oriented Architecture

- The Resource-Oriented Architecture (ROA)
  - Is a RESTful architecture
  - Provides a commonsense set of rules for designing RESTful web services
Resource-Oriented Architecture

- Concepts
  - Resources
    - Resources names (Unified Resource Identifiers-URIs)
    - Resources representations
    - Links between resources

- Key properties:
  - Addressability
  - Statelessness
  - Uniform interface
Resources

- **What’s a Resource?**
  - A resource is any information that
    - can be named
    - Is important enough to be referenced as a thing in itself
  - A resource may be a physical object or an abstract concept
  - e.g.
    - a document
    - a row in a database
    - the result of running an algorithm.
Resources

- Naming:
  - Unified Resource Identifier (URI)
    - The URI is the name and address of a resource
    - Each resource should have at least one URI
    - URIs should have a structure and should vary in predictable ways
Resource

Representation

- A representation is any useful information about the state of a resource

- Different representation formats can be used (Unlike SOAP based Web services)
  - plain-text
  - JSON
  - XML
  - XHTML
  - ....
Resource

... 

- In most RESTful web services, representations are hypermedia
  - i.e. documents that contain data, and links to other resources.
Properties

- **Addressability**
  - An application is addressable if it exposes a URI for every piece of information it serves
  - This may be an infinite number of URIs
    - e.g. for search results
      - [http://www.google.com/search?q=jellyfish](http://www.google.com/search?q=jellyfish)
Properties

- **Statelessness**
  - The state should stay on the client side, and be transmitted to the server for every request that needs it.
    - Makes the protocol simpler
    - Ease load balancing
Properties

- **Uniform interface**
  - **HTTP GET:**
    - Retrieve a representation of a resource
  - **HTTP PUT**
    - Create a new resource, where the client is in charge of creating the resource URI: **HTTP PUT** to the new URI
    - Modify an existing resource: **HTTP PUT** to an existing URI
  - **HTTP POST:**
    - Create a new resource, where the server is in charge of creating the resource URI: **HTTP POST** to the URI of the superordinate of the new resource
  - **HTTP DELETE:**
    - Delete an existing resource:
  - **HTTP HEAD:**
    - Fetch metadata about a resource
  - **HTTP OPTIONS:**
    - Lets the client discover what it’s allowed to do with a resource.
Examples of tool kits

Python
- Django REST

Java
- Play
- Jersey
Examples of RESTful Web Services

- Examples of existing RESTful web services include:
  - Amazon’s Simple Storage Service (S3) (http://aws.amazon.com/s3) (Accessed on September 23, 2023)
  - Oracle Cloud IaaS
The End