

# 09.Services Web

27 avril 2021

## Développement web dlm3

### Services web

HE-Arc (DGR) 2017

## Applications distribuées

- Motivation : répartir l'exécution sur plusieurs machines
  - Principe : Les composants/services communiquent par le réseau
  - Problèmes : Hétérogénéité systèmes, langages, ...
  - Solution : Protocole générique, abstraction différences
  - Exemples : RPC, RMI (java), CORBA, DCOM (MS)
- Utiliser les technologies du web, comme HTTP et XML :
  - indépendantes de la plateforme, éprouvées, largement utilisées
- Système distribué importance de l'architecture :
  - orientée ressource<sup>1</sup> : atome : ressource (donnée) : REST
  - orientée service<sup>2</sup> : atome : service (traitement) : RPC (SOAP)

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Resource-oriented\\_architecture](https://en.wikipedia.org/wiki/Resource-oriented_architecture)

<sup>2</sup>[https://fr.wikipedia.org/wiki/Architecture\\_orient%C3%A9e\\_services](https://fr.wikipedia.org/wiki/Architecture_orient%C3%A9e_services)

## Service web

- 2 visions :
  - Utiliser les technos web pour développer des applis distribuées
  - Accès pour une application aux services offerts aux humains
- Service web = webapp pour une autre application :
  - Webapps : pour humains, via un navigateur (HTTP + HTML)
  - Services web : aux autres applications (HTTP + XML/JSON)
- Exemples :
  - Applications distribuées<sup>3</sup> pour l'entreprise
  - Mashups<sup>4</sup> d'applications web (exemples<sup>5</sup>)
  - Applications Facebook, API Google<sup>6</sup>
  - IFTTT<sup>7</sup>, potions Netvibes<sup>8</sup>
- Consommer un service web ≠ Créer un service web

## SOAP

- AVANT : Simple Object Access Protocol (obsolète)
- Evolution de XML-RPC, format XML d'envoi de messages
- Architecture Orientée Service (SOA)
- Indépendant du langage et de la plateforme
- Recommandation du w3c depuis 2003
- SOAP = abus de langage, service web WS-\* est plus exact
- Spécifications WS-\*<sup>9</sup> :
  - spécifications liées aux différents aspects des services web
  - pour déployer un WS : au minimum SOAP + WSDL + UDDI

## SOAP

- Structure d'un message SOAP

---

<sup>3</sup>[https://upload.wikimedia.org/wikipedia/commons/3/3f/Concept\\_WS.jpg](https://upload.wikimedia.org/wikipedia/commons/3/3f/Concept_WS.jpg)

<sup>4</sup>[https://fr.wikipedia.org/wiki/Application\\_composite](https://fr.wikipedia.org/wiki/Application_composite)

<sup>5</sup><https://www.programmableweb.com/category/mashups>

<sup>6</sup><https://developers.google.com/apis-explorer/>

<sup>7</sup><https://ifttt.com/>

<sup>8</sup><https://www.netvibes.com/fr/explorer/dashboard-of-things>

<sup>9</sup>[https://en.wikipedia.org/wiki/List\\_of\\_web\\_service\\_specifications](https://en.wikipedia.org/wiki/List_of_web_service_specifications)

- Enveloppe, Entête, Corps, Erreurs
- Squelette :

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Header> ... </soap:Header>
  <soap:Body> ...
    <soap:Fault> ... </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

## SOAP

- Exemple<sup>10</sup> requête/réponse
- Introduction à SOAP<sup>11</sup> (fr)
- Créer un service web WS (SOAP) nécessite WSDL et UDDI :
  - SOAP : Echange de messages XML sur le réseau
  - WSDL : Web Service Description Language
  - UDDI : Universal Description, Discovery and Integration
- WSDL : Description des interfaces des web services
- UDDI : Découverte et inscription aux services web
  - annuaire d'informations sur les services web
  - annuaire d'interfaces de services web décrites en WSDL
- Tutorial WSDL/UDDI w3schools<sup>12</sup>

## REST : REpresentational State Transfer

- Style d'architecture sur lequel a été bâti le web
- Architecture Orientée Ressource (ROA)
- Chapitre 5 de la thèse<sup>13</sup> de Roy T. Fielding<sup>14</sup> (fr<sup>15</sup>), 2000

<sup>10</sup>[http://www.w3schools.com/xml/xml\\_soap.asp](http://www.w3schools.com/xml/xml_soap.asp)

<sup>11</sup><http://www.soapuser.com/fr/basics1.html>

<sup>12</sup>[http://www.w3schools.com/xml/xml\\_wsd1.asp](http://www.w3schools.com/xml/xml_wsd1.asp)

<sup>13</sup><http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

<sup>14</sup>[https://fr.wikipedia.org/wiki/Roy\\_Fielding](https://fr.wikipedia.org/wiki/Roy_Fielding)

<sup>15</sup><http://opikanoba.org/tr/fielding/rest/>

- Parmi les contraintes<sup>16</sup>, une interface uniforme :
  - Identification des ressources (URI)
  - Manipulation des ressources par des représentations
  - Messages autodescriptifs
  - Hypermédia comme moteur de l'état de l'application
- Ressource : information ou moyen d'accès
  - ex. : météo du jour, adresse ajout d'un article à un blog, ...
- Représentation : forme donnée à la ressource
  - ex. : page html, fichier PDF, image, flux RSS, fichier sonore, ...

## REST

- Principes
  - Identifier les ressources avec des URI (noms)
  - Actions déterminées par des méthodes HTTP (verbes)
    - \* GET : READ (sûre)
    - \* POST : CREATE
    - \* PUT, PATCH : UPDATE (idempotente)
    - \* DELETE : DELETE (idempotente)
  - Les liens hypertextes permettent de représenter le contenu : navigation
  - Les types MIME déterminent la représentation de la ressource
- Rappel
  - Sûreté : Etat de la ressource (contenu) inchangé
  - Idempotence : plusieurs appels donnent le même résultat

## REST

- L'appel d'une ressource avec des verbes différents produira un résultat différent :

Effet	Route	Verbe	URI (ressource)	Description
C R	Index	GET	/blogs	Affiche la liste
	New	GET	/blog/new	Affiche formulaire création
	Create	POST	/blogs	Création en DB, puis redirection
	Show	GET	/blogs/42	Affiche le blog 42

<sup>16</sup>[https://fr.wikipedia.org/wiki/Representational\\_state\\_transfer](https://fr.wikipedia.org/wiki/Representational_state_transfer)

Effet	Route	Verbe	URI (ressource)	Description
U D	Edit	GET	/blogs/42/edit	Formulaire édition blog 42
	Update	PUT	/blogs/42	MAJ en DB blog 42
	Destroy	DELETE	/blogs/42	Suppression ne DB blog 42

- Laravel, Django, Rails, ... sont RESTful !

## Niveaux de maturité de Richardson<sup>17</sup>

- 0 : Plain Old Xml (POX)
  - Utilisation de HTTP pour faire du RPC
- 1 : Ressources
  - Ressources identifiées par URI
- 2 : Verbes HTTP
  - Respect des propriétés des verbes HTTP
- 3 : Hypertext As The Engine Of Application State (HATEOAS)
  - Les états suivants sont documentés dans la réponse (<link>)

## SOAP vs REST

- webservice : exposer son API en REST ou SOAP ?
- SOAP (WS-\*)
  - hérité du monde de l'entreprise
  - plus de code pour manipuler la requête et générer la réponse
  - plus flexible, extensible (namespace)
  - valider requêtes depuis WDSL
  - nécessité d'un framework (ex : nuSOAP en PHP)
- REST
  - hérité du web
  - plus facile et rapide à utiliser
  - plus lisible et plus compact
  - maintenance plus facile
  - meilleure tolérance aux pannes

<sup>17</sup><http://martinfowler.com/articles/richardsonMaturityModel.html>

## Pour aller plus loin...

- Références
  - SOAP<sup>18</sup>, WSDL<sup>19</sup>, UDDI<sup>20</sup>, XML-RPC<sup>21</sup>, REST<sup>22</sup>, The WSIO<sup>23</sup>
  - Des services web RESTful<sup>24</sup>, Une apologie de REST<sup>25</sup> (recommandés)
  - REST et architectures orientées service<sup>26</sup>, Présentation ROA<sup>27</sup>
  - The RESTful cookbook<sup>28</sup>, Implementing REST<sup>29</sup>
  - How important is HATEOAS<sup>30</sup> (stack overflow)
- Exemples de services web :
  - Google<sup>31</sup>, Yahoo<sup>32</sup>, Flickr<sup>33</sup>, Twitter<sup>34</sup>, Netvibe<sup>35</sup>, ...
  - APIary<sup>36</sup> : Aide au design d'une API REST
- GraphQL<sup>37</sup>
  - est destiné à devenir la prochaine évolution des apis REST utilisant JSON. Initié par Facebook, Github permet également d'en faire usage<sup>38</sup>.

## Sources

---

<sup>18</sup><https://www.w3.org/TR/soap/>

<sup>19</sup><https://www.w3.org/2002/ws/desc/>

<sup>20</sup><http://uddi.xml.org/>

<sup>21</sup><http://xmlrpc.scripting.com/default.html>

<sup>22</sup><http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

<sup>23</sup><http://www.oasis-ws-i.org/>

<sup>24</sup><https://larlet.fr/david/biologeek/archives/20070629-architecture-orientee-ressource-pour-faire-des-services-web-restful/>

<sup>25</sup><https://web.archive.org/web/20160310205502/http://home.ccil.org/~cowan/restws.pdf>

<sup>26</sup><http://www.figer.com/Publications/SOA.htm>

<sup>27</sup><http://fr.slideshare.net/samijaber/symposium-dng-2008-roa>

<sup>28</sup><http://restcookbook.com/>

<sup>29</sup><https://code.google.com/archive/p/implementing-rest/wikis>

<sup>30</sup><http://stackoverflow.com/questions/20335967/how-useful-important-is-rest-hateoas-maturity-level-3>

<sup>31</sup><https://developers.google.com/products/>

<sup>32</sup><https://developer.yahoo.com/everything.html>

<sup>33</sup><https://www.flickr.com/services/api/>

<sup>34</sup><https://dev.twitter.com/overview/api>

<sup>35</sup><http://uwa.netvibes.com/docs/Uwa/html/index.html>

<sup>36</sup><https://apiary.io/>

<sup>37</sup><http://graphql.org/>

<sup>38</sup><https://developer.github.com/v4/>