

09.Services Web

11 octobre 2018

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 ressource¹ : atome : ressource (donnée) : REST
 - orientée service² : atome : service (traitement) : RPC (SOAP)

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)

¹https://en.wikipedia.org/wiki/Resource-oriented_architecture

²https://fr.wikipedia.org/wiki/Architecture_orient%C3%A9e_services

- Exemples :
 - Applications distribuées³ pour l'entreprise
 - Mashups⁴ d'applications web (exemples⁵)
 - Applications Facebook, API Google⁶
 - IFTTT⁷, potions Netvibes⁸
- 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-*⁹ :
 - spécifications liées aux différents aspects des services web
 - pour déployer un WS : au minimum SOAP + WSDL + UDDI

SOAP

- Structure d'un message SOAP
 - 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>
```

³https://upload.wikimedia.org/wikipedia/commons/3/3f/Concept_WS.jpg

⁴https://fr.wikipedia.org/wiki/Application_composite

⁵<http://www.programmableweb.com/category/all/mashups>

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

⁷<https://ifttt.com/>

⁸<http://www.netvibes.com/fr/dashboardofthings>

⁹https://en.wikipedia.org/wiki/List_of_web_service_specifications

SOAP

- Exemple¹⁰ requête/réponse
- Introduction à SOAP¹¹ (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 w3schools¹²

REST : REpresentational State Transfer

- Style d'architecture sur lequel a été bâti le web
- Architecture Orientée Ressource (ROA)
- Chapitre 5 de la thèse¹³ de Roy T. Fielding¹⁴ (fr¹⁵), 2000
- Parmi les contraintes¹⁶, 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)

¹⁰http://www.w3schools.com/xml/xml_soap.asp

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

¹²http://www.w3schools.com/xml/xml_wsd.asp

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

¹⁴https://fr.wikipedia.org/wiki/Roy_Fielding

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

¹⁶https://fr.wikipedia.org/wiki/Representational_state_transfer

- * 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
 - Sécurité : Etat de la ressource (contenu) inchangé
 - Idempotence : plusieurs appels donnent le même résultat

REST

- URI logique plutôt qu'URL physique
- L'appel d'une ressource avec des méthodes différentes produira un résultat différent :

```
* GET      http://www.monblog.com/posts    // Liste des billets
* GET      http://www.monblog.com/posts/1  // Billet 1
* POST     http://www.monblog.com/posts    // Création d'un billet
* PUT/PATCH http://www.monblog.com/posts/1 // Mise à jour billet 1
* DELETE   http://www.monblog.com/posts/1 // Suppr billet 1
```

- Avec Laravel¹⁷ ou Rails, ces actions sont nommées : index, show, store/create, update, destroy
- Laravel et Rails sont RESTful !

Niveaux de maturité de Richardson¹⁸

- 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>)

¹⁷<https://laravel.com/docs/master/controllers#resource-controllers>

¹⁸<http://martinfowler.com/articles/richardsonMaturityModel.html>

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

Pour aller plus loin...

- Références
 - SOAP¹⁹, WSDL²⁰, UDDI²¹, XML-RPC²², REST²³, The WSIO²⁴
 - Des services web RESTful²⁵, Une apologie de REST²⁶ (recommandés)
 - REST et architectures orientées service²⁷, Présentation ROA²⁸
 - The RESTful cookbook²⁹, Implementing REST³⁰
 - How important is HATEOAS³¹ (stack overflow)
- Exemples de services web :
 - Google³², Yahoo³³, Flickr³⁴, Twitter³⁵, Netvibe³⁶, ...

¹⁹<https://www.w3.org/TR/soap/>

²⁰<https://www.w3.org/2002/ws/desc/>

²¹<http://uddi.xml.org/>

²²<http://xmlrpc.scripting.com/default.html>

²³<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

²⁴<http://www.oasis-ws-i.org/>

²⁵<https://larlet.fr/david/biologeeek/archives/20070629-architecture-orientee-ressource-pour-faire-des-services-web-restful/>

²⁶<https://web.archive.org/web/20160310205502/http://home.ccil.org/~cowan/restws.pdf>

²⁷<http://www.figer.com/Publications/SOA.htm>

²⁸<http://fr.slideshare.net/samijaber/symposium-dng-2008-roa>

²⁹<http://restcookbook.com/>

³⁰<https://code.google.com/archive/p/implementing-rest/wikis>

³¹<http://stackoverflow.com/questions/20335967/how-useful-important-is-rest-hateoas-maturity-level-3>

³²<https://developers.google.com/products/>

³³<https://developer.yahoo.com/everything.html>

³⁴<https://www.flickr.com/services/api/>

³⁵<https://dev.twitter.com/overview/api>

³⁶<http://uwa.netvibes.com/docs/Uwa/html/index.html>

- APIary³⁷ : Aide au design d'une API REST
- GraphQL³⁸
 - est destiné à devenir la prochaine évolution des apis REST utilisant JSON. Initié par Facebook, Github permet également d'en faire usage³⁹.

Sources

³⁷<https://apiary.io/>

³⁸<http://graphql.org/>

³⁹<https://developer.github.com/early-access/graphql/>