09.Services Web

29 janvier 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 ressource[1] : atome : ressource (donnée) : REST
  - orientée service[2] : 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)

[1] https://en.wikipedia.org/wiki/Resource-oriented_architecture
[2] https://fr.wikipedia.org/wiki/Architecture_orient%C3%A9e_services

- Exemples :
  - Applications distribuées[3] pour l'entreprise
  - Mashups[4] d'applications web (exemples[5])
  - Applications Facebook, API Google[6]
  - IFTTT[7], potions Netvibes[8]
- 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>
```

[3] https://upload.wikimedia.org/wikipedia/commons/3/3f/Concept_WS.jpg
[4] https://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
[9] https://en.wikipedia.org/wiki/List_of_web_service_specifications

SOAP

- Exemple[10] requête/réponse
- Introduction à SOAP[11] (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[12]

REST : REpresentational State Transfer

- Style d'architecture sur lequel a été bâti le web
- Architecture Orientée Ressource (ROA)
- Chapitre 5 de la thèse[13] de Roy T. Fielding[14] (fr[15]), 2000
- Parmi les contraintes[16], 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)

[10] http://www.w3schools.com/xml/xml_soap.asp
[11] http://www.soapuser.com/fr/basics1.html
[12] http://www.w3schools.com/xml/xml_wsdl.asp
[13] http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
[14] https://fr.wikipedia.org/wiki/Roy_Fielding
[15] http://opikanoba.org/tr/fielding/rest/
[16] https://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 determinent la représentation de la ressource
- Rappel
    - Sûreté : 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 Laravel[17] ou Rails, ces actions sont nommées : `index`, `show`, `store/create`, `update`, `destroy`
- Laravel et Rails sont RESTful !

Niveaux de maturité de Richardson[18]

- 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
  - SOAP[19], WSDL[20], UDDI[21], XML-RPC[22], REST[23], The WSIO[24]
  - Des services web RESTful[25], Une apologie de REST[26] (recommandés)
  - REST et architectures orientées service[27], Présentation ROA[28]
  - The RESTful cookbook[29], Implementing REST[30]
  - How important is HATEOAS[31] (stack overflow)
- Exemples de services web :
  - Google[32], Yahoo[33], Flickr[34], Twitter[35], Netvibe[36], ...

[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/biologeek/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

  - APIary[37] : Aide au design d'une API REST
- GraphQL[38]
  - est destiné à devenir la prochaine évolution des apis REST utilisant JSON. Initié par Facebook, Github permet également d'en faire usage[39].

Sources

[37] https://apiary.io/
[38] http://graphql.org/
[39] https://developer.github.com/early-access/graphql/