09.Services Web

31 janvier 2024

Développement web il3

Services web

HE-Arc (DGR) 2022

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)

[1] https://en.wikipedia.org/wiki/Resource-oriented_architecture
[2] 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[3] pour l'entreprise
  - Mashups[4] d'applications web (exemples[5])
  - Applications Facebook, API Google[6]
  - IFTTT[7]
- 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-*[8] :
  - 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

[3] https://upload.wikimedia.org/wikipedia/commons/3/3f/Concept_WS.jpg
[4] https://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)
[5] https://science.howstuffworks.com/innovation/repurposed-inventions/5-web-mashups.htm
[6] https://developers.google.com/apis-explorer/
[7] https://ifttt.com/
[8] https://en.wikipedia.org/wiki/List_of_web_service_specifications

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

REST : REpresentational State Transfer

- Style d'architecture sur lequel a été bâti le web
- Architecture Orientée Ressource (ROA)
- Chapitre 5 de la thèse[12] de Roy T. Fielding[13] (fr[14]), 2000
- Parmi les contraintes[15], une interface uniforme :

[9] https://www.w3schools.com/xml/xml_soap.asp
[10] https://www.w3big.com/fr/soap/default.html#gsc.tab=0
[11] https://www.w3schools.com/xml/xml_wsdl.asp
[12] https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
[13] https://fr.wikipedia.org/wiki/Roy_Fielding
[14] https://opikanoba.org/tr/fielding/rest/
[15] https://fr.wikipedia.org/wiki/Representational_state_transfer

  - 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 determinent 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
	Index	GET	/blogs	Affiche la liste
	New	GET	/blog/new	Affiche formulaire création
C	Create	POST	/blogs	Création en DB, puis redirection
R	Show	GET	/blogs/42	Affiche le blog 42
	Edit	GET	/blogs/42/edit	Formulaire édition blog 42
U	Update	PUT	/blogs/42	MAJ en DB blog 42
D	Destroy	DELETE	/blogs/42	Suppression ne DB blog 42

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

Niveaux de maturité de Richardson[16]

- 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

Pour aller plus loin...

- Références

[16] https://martinfowler.com/articles/richardsonMaturityModel.html

  - SOAP[17], WSDL[18], UDDI[19], REST[20], The WSIO[21]
  - Des services web RESTful[22], Une apologie de REST[23] (recommandés)
  - REST et architectures orientées service[24], Présentation ROA[25]
  - The RESTful cookbook[26], How important is HATEOAS[27] (stack overflow)

- Exemples de services web :
  - Google[28], Yahoo[29], Flickr[30], Twitter[31], ...
  - APIary[32] : Aide au design d'une API REST
  - Tests : Postman, Hoppscotch[33], Ping-API[34], autres[35]

- GraphQL[36]
  - est destiné à devenir la prochaine évolution des apis REST utilisant JSON. Initié par Facebook, Github permet également d'en faire usage[37].

Sources

[17] https://www.w3.org/TR/soap/
[18] https://www.w3.org/2002/ws/desc/
[19] https://uddi.xml.org/
[20] https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
[21] https://www.oasis-ws-i.org/
[22] https://larlet.fr/david/biologeek/archives/20070629-architecture-orientee-ressource-pour-faire-des-services-web-restful/
[23] https://web.archive.org/web/20160310205502/http://home.ccil.org/~cowan/restws.pdf
[24] https://www.figer.com/Publications/SOA.htm
[25] https://fr.slideshare.net/samijaber/symposium-dng-2008-roa
[26] https://restcookbook.com/
[27] https://stackoverflow.com/questions/20335967/how-useful-important-is-rest-hateoas-maturity-level-3
[28] https://developers.google.com/products/
[29] https://developer.yahoo.com/everything.html
[30] https://www.flickr.com/services/api/
[31] https://dev.twitter.com/overview/api
[32] https://apiary.io/
[33] https://hoppscotch.io/
[34] https://ping-api.com/
[35] https://testsigma.com/blog/postman-alternatives/
[36] http://graphql.org/
[37] https://developer.github.com/v4/