

09.Services Web

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

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

²https://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 distribuées³ pour l'entreprise
 - Mashups⁴ d'applications web (exemples⁵)
 - Applications Facebook, API Google⁶
 - IFTTT⁷
- 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

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

⁴[https://en.wikipedia.org/wiki/Mashup_\(web_application_hybrid\)](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/>

⁸https://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

- Exemple⁹ requête/réponse
- Introduction à SOAP¹⁰ (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 w3schools¹¹

REST : REpresentational State Transfer

- Style d'architecture sur lequel a été bâti le web
- Architecture Orientée Ressource (ROA)
- Chapitre 5 de la thèse¹² de Roy T. Fielding¹³ (fr¹⁴), 2000
- Parmi les contraintes¹⁵, une interface uniforme :

⁹https://www.w3schools.com/xml/xml_soap.asp

¹⁰<https://www.w3big.com/fr/soap/default.html#gsc.tab=0>

¹¹https://www.w3schools.com/xml/xml_wsdl.asp

¹²<https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

¹³https://fr.wikipedia.org/wiki/Roy_Fielding

¹⁴<https://opikanoba.org/tr/fielding/rest/>

¹⁵https://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 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	Index	GET	/blogs	Affiche la liste
	New	GET	/blog/new	Affiche formulaire création
	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 Richardson¹⁶

- 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>

- SOAP¹⁷, WSDL¹⁸, UDDI¹⁹, REST²⁰, The WSIO²¹
- Des services web RESTful²², Une apologie de REST²³ (recommandés)
- REST et architectures orientées service²⁴, Présentation ROA²⁵
- The RESTful cookbook²⁶, How important is HATEOAS²⁷ (stack overflow)
- Exemples de services web :
 - Google²⁸, Yahoo²⁹, Flickr³⁰, Twitter³¹, ...
 - APIary³² : Aide au design d'une API REST
 - Tests : Postman, Hoppscotch³³, Ping-API³⁴, autres³⁵
- GraphQL³⁶
 - est destiné à devenir la prochaine évolution des apis REST utilisant JSON. Initié par Facebook, Github permet également d'en faire usage³⁷.

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/biologeeek/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/>