R4.01 - Web Services

1 - Définitions

Web Service

Un Web Service (WS) est un moyen d'échanger des informations entre 2 systèmes informatiques, un client et un serveur par exemple, en utilisant :

- les technologies du Web (serveur Web, protocole HTTP)
- des règles de dialogue standardisées (protocole, format JSON, XML, HTML…)

Web Service

Un WS :

- Joue un rôle de fournisseur de service
- Propose une API pour accéder à ce service, uniquement sur un réseau (privé ou public)
- Est accédé, via l'API, par des applications qui jouent un rôle de consommateurs du service

Fournisseur & Consommateur

Un WS n'impose aucune finalité. Il doit simplement se conformer à un protocole Internet (généralement HTTP/HTTPs) et fournir un service.

Un serveur Web qui délivre des pages HTML peut donc être considéré comme un fournisseur de WS !

Un navigateur Web peut alors être le consommateur.

Fournisseur & Consommateur

Dans l'exemple précédent d'un serveur Web qui délivrerait des pages HTML, ce service peut se résumer simplement en :

"fournisseur de page HTML"

Cependant, un WS va généralement plus loin que ça !

Fournisseur & Consommateur

Dans le cas particulier des WS, derrière le terme service il faut comprendre, très souvent, qu'il y a une histoire de base de données.

L'API permet de manipuler les données de cette base.

API d'un Web Service

Une API (Application Programming Interface) :

- Constitue un ensemble d'actions de manipulation des données fournies par le WS
- Les actions sont appelées des verbes.
- Les données sont appelées des ressources.

Ressources

Les ressources sont des types de données.

Une ressource est représentée par une URL.

Exemples :

- http://localhost/ws/clients : cible tous les clients
- http://localhost/ws/clients/123 : cible le client 123

Actions

Un WS met à disposition des consommateurs tout ou partie d'un lot d'actions (verbes) sur des ressources.

Ces actions sont connues sous l'acronyme CRUD :

- Create (INSERT en SQL) - Création d'une ressource
- Read (SELECT) - Lecture d'une ressource
- Update (UPDATE) - Modification d'une ressource
- Delete (DELETE) - Suppression d'une ressource

Verbes

Voici les noms des verbes du protocole pour chacune des actions d'une API WS :

- Création : POST
- Récupération (lecture) : GET
- Mise à jour complète (tous les champs) : PUT
- Mise à jour partielle (certains champs) : PATCH
- Suppression : DELETE

Verbes

Un navigateur Web sait utiliser seulement 2 verbes :

- GET : pour lire une ressource = demander une page HTML au serveur (ou d'autres éléments comme les images, le CSS, des scripts JS etc.)
- POST : pour envoyer au serveur des données, généralement issues d'un formulaire.

Verbes

C'est moins commun, mais rien n'empêche de créer d'autres verbes pour une API. Suggestions :

- CLONE : pour dupliquer une ressource
- ARCHIVE : pour rendre indisponible une ressource sans la supprimer.

Le choix est libre pour le nommage d'autres verbes, mais il faut bien documenter ce qui sort de l'ordinaire !

2 - Réponses

Codes HTTP

Un WS s'appuie sur un protocole qui est souvent HTTP.

Chaque réponse du WS est associée à un code.
Exemples :

- 200 - OK - Tout s'est bien passé
- 404 - Not found - La ressource n'existe pas
- 403 - Forbidden - Accès refusé

Liste : https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP

Format du BODY

Une réponse peut aussi être associée à des données renvoyées au client (consommateur), dans le BODY de la réponse.

Généralement c'est un GET qui renvoie des données mais d'autres verbes peuvent décider d'en renvoyer aussi, même si un simple code est souvent suffisant.

Format du BODY

Le BODY peut être formaté comme on veut.

C'est généralement du texte, éventuellement compressé (gzip)

Des formats standards sont généralement utilisés : JSON, XML.

3 - Familles de WS

Il y a globalement 2 familles de WS :

- REST
- SOAP

REST est le plus simple et le plus accessible.
Ils diffèrent sur des aspects techniques qu'on abordera pas ici, mais ils sont quand même très similaires.

4 - Précisions

Chemins aux ressources

Les ressources sont ciblées par des URL.
Une URL doit “mener” directement à la ressource ou à un ensemble de ressources de même type.

Rappel des 2 exemples précédents :

- http://localhost/ws/clients : cible tous les clients
- http://localhost/ws/clients/123 : cible le client 123

Chemins aux ressources

Les actions ciblent directement des ressources et ne doivent pas introduire autre chose dans le “chemin vers les ressources”. Exemples :

GET http://localhost/ws/clients -> OK
GET http://localhost/ws/clients/123 -> OK
GET http://localhost/ws/clients/pays/france -> KO !

Chemins aux ressources

L'idée derrière l'URL de la ressource est la même que celle pour accéder à un fichier dans une arborescence.

On ne dirait pas :
cat /clients/123.txt/domicilié_en_france
si le fichier s'appelle simplement 123.txt même s'il y a son adresse dans le fichier texte !
C'est donc pareil avec les ressources des WS.

Paramètres

Les actions qui nécessitent des paramètres peuvent spécifier ces paramètres dans l'URL ainsi :

GET http://localhost/ws/clients?pays=france -> OK !

Le chemin vers la ressource est tout ce qui se trouve AVANT le ?. Les paramètres multiples doivent être séparés par des &. Exemple : ?pays=france&dept=22

Exemples

Quelques exemples d'actions :

- GET /api/v1/employes : Récupération de tous les employés
- GET /api/v1/employes/5 : Récupération de l'employé n°5
- POST /api/v1/employes : Création d'un nouvel employé
- DELETE /api/v1/employes/12 : Suppression de l'employé n°12
- PUT /api/v1/employes/15 : Modification de l'employé n°15

https://commons.wikimedia.org/wiki/File:Thats_all_folks.svg