

API REST

MORISSEAU Mickael

API

Qu'est ce qu'une API ?

Qu'est ce qu'une API

API pour Application Programming Interfaces

Besoin :

- Permet de communiquer entre applications
- Ne pas tout redévelopper à chaque fois

Exemple :

- Mire de connexion (via Formulaire, Facebook, Google, ...)
- Payement sur les sites (banque, PayPal, ...)

Qu'est ce qu'une API

Dans la vie de tous les jours :

- Télécommande
 - Interface entre l'humain et le matériel



Qu'est ce qu'une API

Dans la vie de tous les jours :

- Télécommande
 - Interface entre l'humain et le matériel

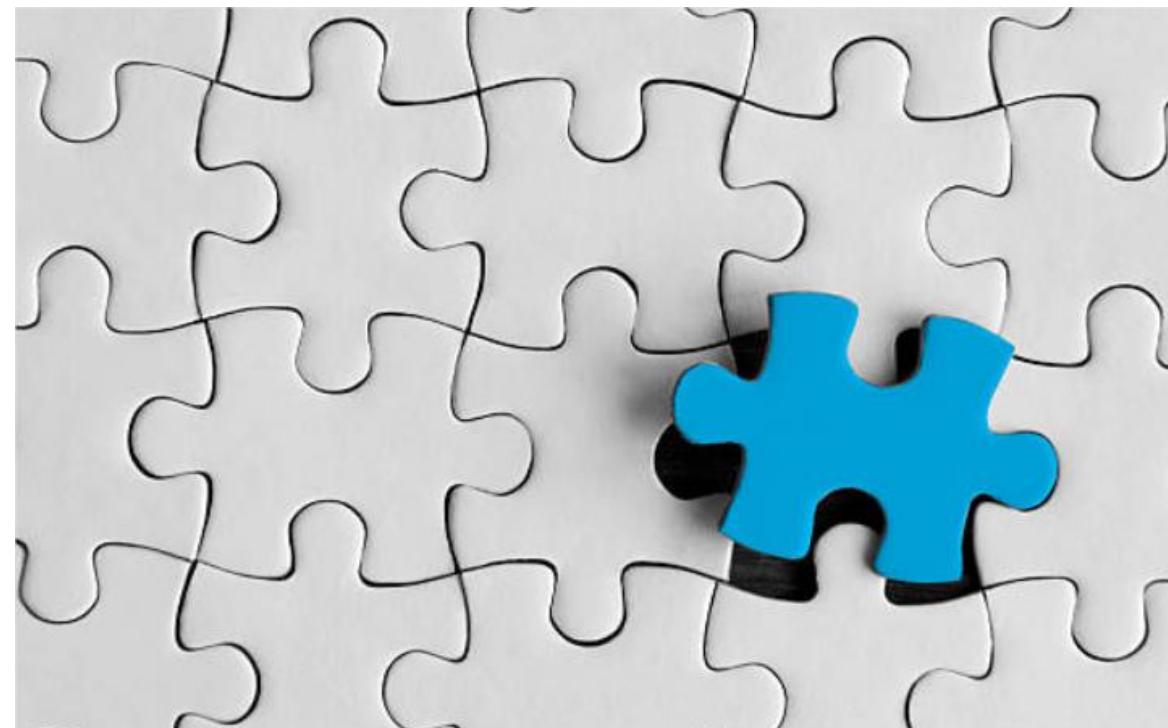


Qu'est ce qu'une API

Nous utilisons beaucoup d'interfaces sans nous en rendre compte.

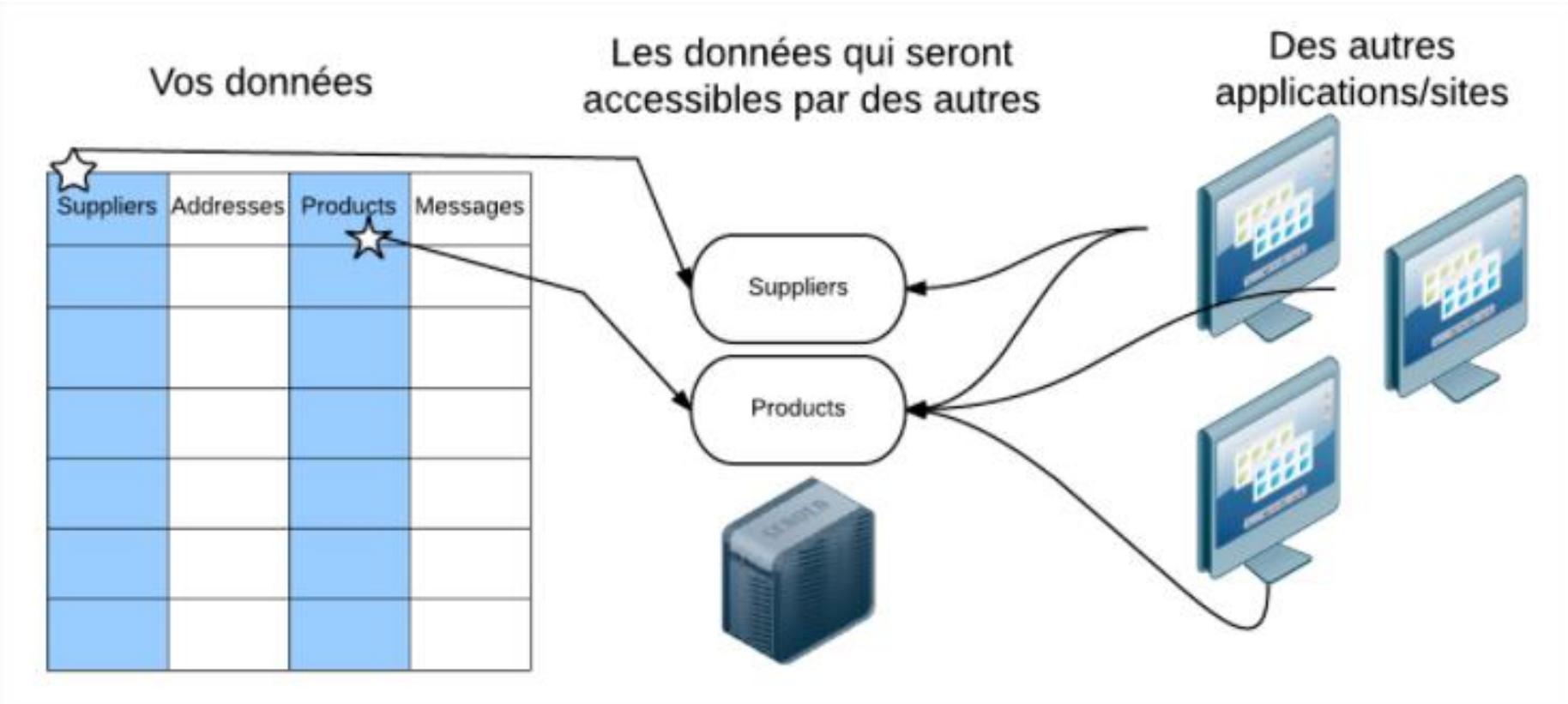
Les sites web et les applications ont besoin de la même chose pour communiquer et échanger des données.

Une API est une interface pour les applications.



Qu'est ce qu'une API

Créer sa propre API, pourquoi ?



REST

Qu'est ce que REST ?

Qu'est ce que REST

REST pour REpresentational State Transfer

Il s'agit d'une style d'architecture pour les systèmes hypermédia distribués.

Créé par Roy Fielding en 2000, informaticien Américain, dans le chapitre 5 de sa thèse de doctorat1 ("Architectural Styles and the Design of Network-based Software Architectures").

Il a beaucoup contribué à l'informatique, notamment des applications dans le **World Wide Web** (W3C), comme :

HTTP

HTML

URI

Apache (cofondateur de la fondation Apache)

...

Il continue d'écrire sur ce concept, et une grande communauté de personnes contribue à maintenir les standards des API REST à un haut niveau.

Qu'est ce que REST

L'explication telle que donnée par Roy T. Fielding, de la signification de REST est la suivante :

"Representational State Transfer évoque l'image du fonctionnement d'une application Web bien construite : un réseau de pages Web (une machine à états finis virtuelle) où l'utilisateur progresse dans l'application en cliquant sur des liens (transition entre états) ce qui provoque l'affichage de la page suivante (représentant le nouvel état de l'application) à l'utilisateur qui peut alors l'exploiter".



Qu'est ce que REST

L'architecture REST a pris de plus en plus d'importance ces dernières années, et on retrouve ce type d'architecture dans de nombreux services et réseaux sociaux.

Lors de la conception ou évolution, il est maintenant de plus en plus difficile d'imaginer un service qui ne propose pas d'API avec laquelle interagir.

Cela change à la fois l'écosystème auquel nous sommes habitué mais aussi cela modifie notre manière de concevoir des applications.

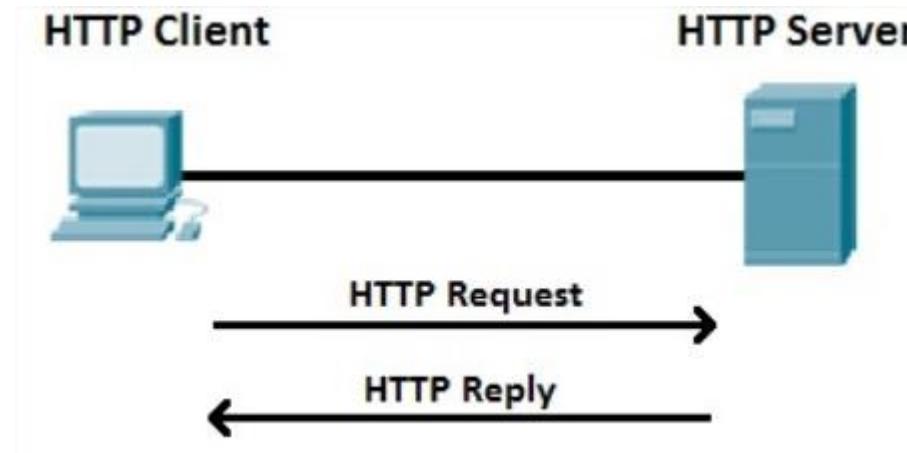
Attention :

Le REST est un style d'architecture logiciel, c'est l'architecture originelle du web et non un standard.

REST est basé sur le HTTP

Les API REST sont basées sur HTTP (pour Hypertext Transfer Protocol)

C'est un protocole qui définit la communication entre les différentes parties du web.
L'échange est basé sur des requêtes client et serveur. Un client lance une requête HTTP, et le serveur renvoie une réponse.



Attention : Toutes les API ne sont pas basées sur HTTP

Particularité du REST

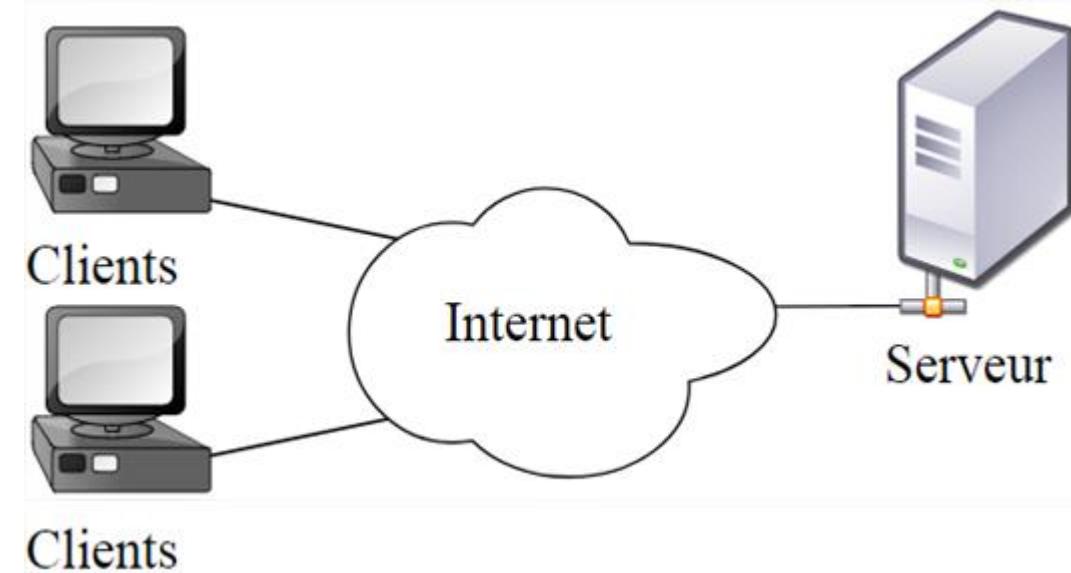
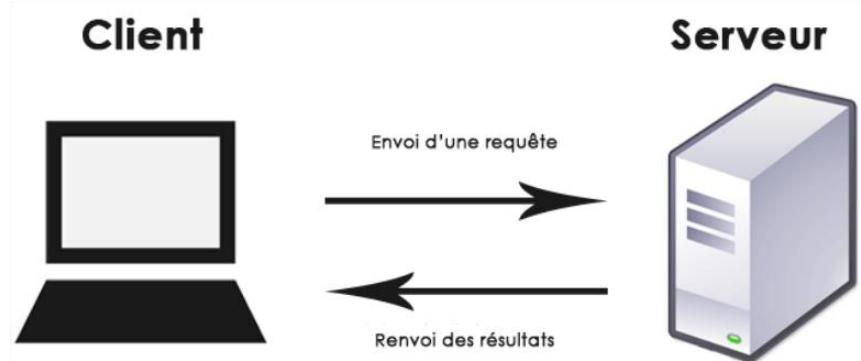
Les API REST imitent la façon dont le web lui-même marche dans les échanges entre un client et un serveur. Une API REST est :

- Sans état
- Cacheable (avec cache = mémoire)
- Orienté client-serveur
- Avec une interface uniforme
- Avec un système de couche
- Un code à la demande (optionnel)

Particularité du REST

Le principe du client-serveur définit les deux entités qui interagissent dans une API REST : un **client** et un **serveur**, les mêmes entités qui communiquent sur le web.

Un **client** envoie une requête et le **serveur** renvoie une réponse.



Les réponses du serveur pour les API REST peuvent être délivrées dans de multiples formats, les plus utilisés :

JSON (JavaScript Object Notation) est souvent utilisé, mais XML, CSV, ou même RSS sont aussi valables.

Standards utilisé par REST

URI (Uniform Ressource Identifier) comme syntaxe universelle pour adresser les ressources,

HTTP un protocole sans état (**stateless**) avec un nombre très limité d'opérations,

Des **liens hypermedia** dans des documents (X)HTML et XML pour représenter à la fois le contenu des informations et la transition entre états de l'application,

Les **types MIME** comme text/xml, text/html, image/jpeg, application/pdf, video/mpeg, etc pour la représentation des ressources.

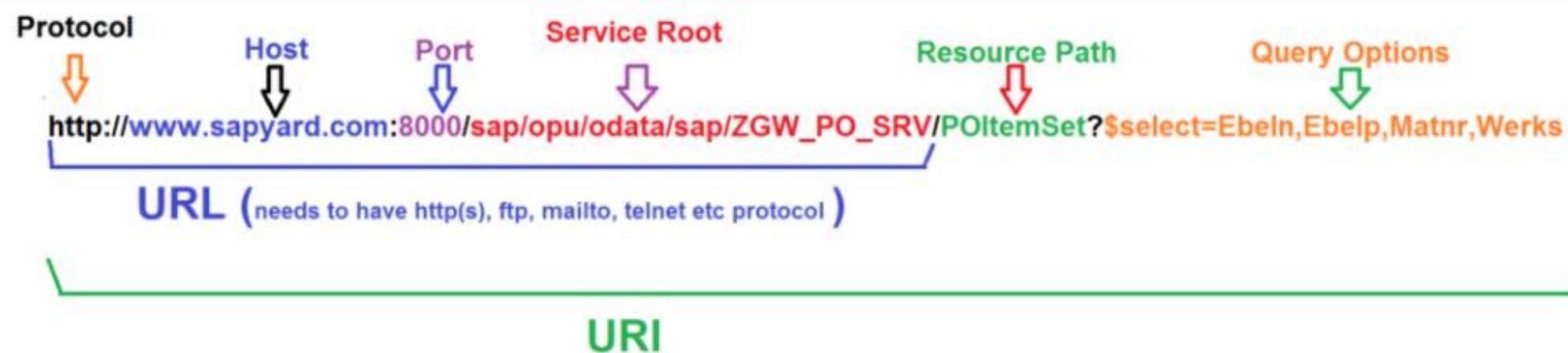
REST concerne l'architecture globale d'un système. Il ne définit pas la manière de réaliser dans les détails. En particulier, des services REST peuvent être réalisés en .NET, JAVA, CGI ou COBOL.

[Complément] URI, URL, ...

`http://www.sapyard.com:8000/sap/opu/odata/sap/ZGW_PO_SRV/POItemSet?$select=Ebeln,Ebelp,Matnr,Werks`

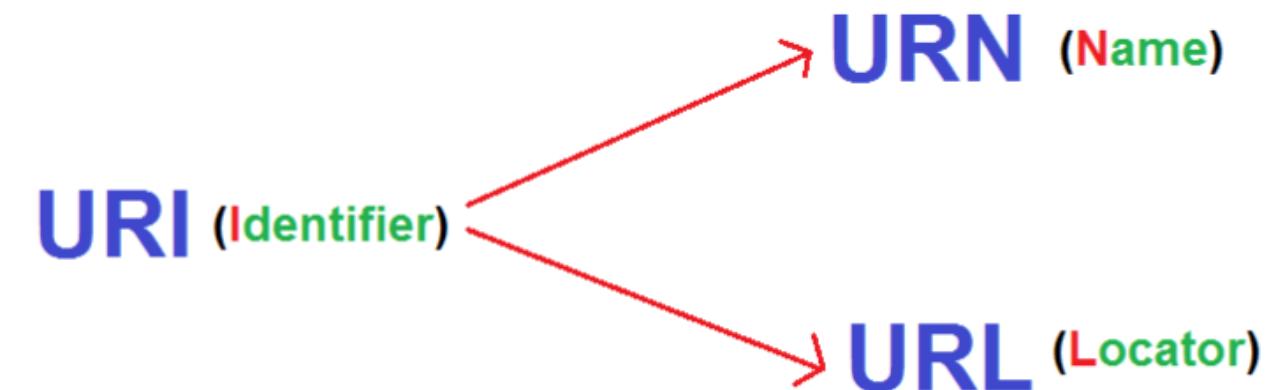
- Protocol
- Host
- Port
- Racine du service
- Ressource
- Option(s)
- URL
- URI
- URN

[Complément] URI, URL, ...



Simplement :

- URL : un nom
- URI : un localisateur
- URN : un identifiant



Standards utilisé par REST

La ressource :

Dans REST, la plus important c'est la ressource. On utilise une ressource, ou plutôt sa représentation. La ressource est "quelque chose" que l'on créé, qui va évoluer avec le temps. On peut la modifier et la détruire via un "composant". Cette représentation est une séquence d'octet et est éventuellement accompagnée de métadonnées.

Le composant :

Le composant est un acteur qui agit sur une ressource via un canal. Chaque composant peut être lié à plusieurs ressources et plusieurs autres composants. Ses liaisons permettent des interactions sans états.

Les types MIME :

Le Content-Type ou MIME Type est très important puisque c'est lui qui est utilisé par le navigateur pour déterminer le type du fichier, pas l'extension. C'est en fonction de ce type de fichier que le navigateur déterminera l'action à accomplir (visualisation, téléchargement, etc..). Lorsque la même ressource existe sous plusieurs représentations ou plusieurs langages, il est aussi possible à l'agent utilisateur de "négocier" avec le serveur la représentation qui sera fournie.

Standards utilisé par REST

URI :

L'uri est une courte chaîne de caractère qui va permettre d'identifier une ressource de manière permanente sur un réseau, physique ou abstraite, même si la ressource est déplacée ou supprimée. Connaître l'URI suffit pour agir sur la ressource.

Note :

Cette notion d'URI est fondamentale, car c'est le système global et unique d'identification du Web. L'URI est la pierre angulaire de l'architecture Web. Le système d'URI a été largement déployé depuis les débuts du Web.

Les avantages des URI sont nombreux : liens, favoris, mécanismes de cache, indexation par les moteurs de recherches.

En utilisant les URI, il est possible de déployer une application partout dans le monde sans infrastructure additionnelle comme des annuaires ou des "registries". Déployer un autre système de nommage qui aurait les mêmes propriétés que les URI serait très coûteux.

Standards utilisé par REST

HTTP :

HTTP va fournir les opération nécessaires à la manipulation de la ressource :

- GET pour récupérer la ressource
- POST pour créer la ressource
- PUT pour modifier la ressource
- DELETE pour supprimer la ressource.

Standards utilisé par REST – requêtes

GET :

GET est la méthode la plus utilisée pour les requêtes HTTP.

Comme le suggère le mot “GET” en anglais, cette méthode existe pour récupérer des données d'une ressource.

POST :

POST permet d'envoyer des données dans une requête et souvent pour l'ajouter à la ressource précisée dans la partie URI de la première ligne de la requête.

Le type de contenu dans le corps de la requête peut être défini avec un en-tête « Content-type » pour que le serveur sache comment traiter les données dans la requête.

Standards utilisé par REST – requêtes

PUT :

PUT est peu utilisé.

Contrairement à POST, PUT indique que seul l'URI précisé par le client doit être affecté, point final.

DELETE :

DELETE permet de supprimer la ressource donnée dans l'URI.

Note : Par contre, vous ne pourrez pas être sûr que l'action s'est vraiment passée, car la réponse du serveur, quelle qu'elle soit, n'est pas une véritable confirmation de suppression.

Standards utilisé par REST

Réponse :

Code réponse HTTP :

Indique l'état de la réponse.

Les codes HTTP sont toujours trois chiffres et

Ils sont catégorisés en par le chiffre des centaines

Un code de la forme :

1xx indique une information.

2xx indique le succès.

3xx redirige le client ailleurs.

4xx indique une faute de la part du client.

5xx indique une erreur de la part du serveur.

Exemple :

404 => erreur, page non trouvée

200 => succès de la requête

401 => erreur, utilisateur non authentifié

403 => erreur, accès refusé

500 et 503 => erreur serveur

Le corps :

Il s'agit de la partie la plus importante.

Le corps de la réponse contient les données que l'on veut obtenir (suite à l'appel).

Différence entre JSON, XML et CSV

JSON :

```
{"messages": [{"courses": [{"numero": "9206",
  "dateCirculation": "2016-12-09",
  "horaireDepart": "06:34",
  "horaireDestination": "10:37",
  "lieuOrigine": {"uic": "85030007",
    "libPr": "Zürich HB"}, "lieuDestination": {"uic": "87686006",
    "libPr": "Paris-Gare-de-Lyon"}}], "libPr": "Paris-Gare-de-Lyon"}]}
```

Différence entre JSON, XML et CSV

JSON :

```
{"messages": [{"courses": [{"numero": "9206", "dateCirculation": "2016-12-09", "horaireDepart": "06:34", "horaireDestination": "10:37", "lieuOrigine": {"uic": "85030007", "libPr": "Zürich HB"}, "lieuDestination": {"uic": "87686006", "libPr": "Paris-Gare-de-Lyon"}}], "]}]
```

XML :

```
<?xml version="1.0" encoding="UTF-8" ?>
<messages>
    <courses>
        <numero>9206</numero>
        <dateCirculation>2016-12-09</dateCirculation>
        <horaireDepart>06:34</horaireDepart>
        <horaireDestination>10:37</horaireDestination>
        <lieuOrigine>
            <uic>85030007</uic>
            <libPr>Zürich HB</libPr>
        </lieuOrigine>
        <lieuDestination>
            <uic>87686006</uic>
            <libPr>Paris-Gare-de-Lyon</libPr>
        </lieuDestination>
    </courses>
</messages>
```

Différence entre JSON, XML et CSV

JSON :

```
{"messages": [{"courses": [{"numero": "9206", "dateCirculation": "2016-12-09", "horaireDepart": "06:34", "horaireDestination": "10:37", "lieuOrigine": {"uic": "85030007", "libPr": "Zürich HB"}, "lieuDestination": {"uic": "87686006", "libPr": "Paris-Gare-de-Lyon"}}], "]}]
```

XML :

```
<?xml version="1.0" encoding="UTF-8" ?>
<messages>
    <courses>
        <numero>9206</numero>
        <dateCirculation>2016-12-09</dateCirculation>
        <horaireDepart>06:34</horaireDepart>
        <horaireDestination>10:37</horaireDestination>
        <lieuOrigine>
            <uic>85030007</uic>
            <libPr>Zürich HB</libPr>
        </lieuOrigine>
        <lieuDestination>
            <uic>87686006</uic>
            <libPr>Paris-Gare-de-Lyon</libPr>
        </lieuDestination>
    </courses>
</messages>
```

CSV :

numero;dateCirculation;horaireDepart;horaireDestination;lieuOrigineUIC;lieuOrigineLibPr;lieuDestinationUIC;lieuDestinationLibPr
9206;2016-12-09;06:34;10:37;85030007;Zürich HB;87686006;Paris-Gare-de-Lyon

Différence entre JSON, XML et CSV

Exercice :

Création d'une BDD sur le model de donnée présentée

Rappel du CSV :

numero;dateCirculation;horaireDepart;horaireDestination;lieuOrigineUIC;lieuOrigineLibPr;lieuDestinationUIC;lieuDestinationLibPr
9206;2016-12-09;06:34;10:37;85030007;Zürich HB;87686006;Paris-Gare-de-Lyon

Structure (possible) :

#	Nom	Type
1	numero	int(11)
2	dateCirculation	varchar(10)
3	horaireDepart	varchar(5)
4	horaireDestination	varchar(5)
5	lieuOrigineUIC	varchar(8)
6	lieuOrigineLibPr	varchar(255)
7	lieuDestinationUIC	varchar(8)
8	lieuDestinationLibPr	varchar(255)

Différence entre JSON, XML et CSV

BDD :

	Colonnes	Données	Informations d'index	Contraintes				
1	numero	dateCirculation	horaireDepart	horaireDestination	lieuOrigineUIC	lieuOrigineLibPr	lieuDestinationUIC	lieuDestinationLibPr

Actions... ▾

	numero	dateCirculation	horaireDepart	horaireDestination	lieuOrigineUIC	lieuOrigineLibPr	lieuDestinationUIC	lieuDestinationLibPr
1	9206	2016-12-09	06:34	10:37	85030007	Zürich HB	87686006	Paris-Gare-de-Lyon

SERVICE WEB

Qu'est ce qu'un service Web ?

Service Web (WebService)

La technologie des services Web est un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates-formes. Les services Web sont basés sur le modèle SOA .

Intérêt :

Les services Web fournissent un lien entre applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde. ;)

Les services Web sont normalisés, car ils utilisent les standards XML et HTTP pour transférer des données et ils sont compatibles avec de nombreux autres environnements de développement.

Ils sont donc indépendants des plates-formes.

Avantage :

Permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises.

Il n'imposent pas de modèles de programmation spécifiques.

Service Web (WebService)

Caractéristiques :

La technologie des services Web repose essentiellement sur une représentation standard des données (interfaces, messageries).

Cette technologie est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web.

Un service Web possède les caractéristiques suivantes :

- il est accessible via le réseau ;
- il dispose d'une interface publique (ensemble d'opérations) ;
- il communique, ces messages sont transportés par des protocoles Internet (généralement HTTP, mais rien n'empêche d'utiliser d'autres protocoles de transfert tels : SMTP, FTP, BEEP...) ;
- l'intégration d'application en implémentant des services Web produit des systèmes faiblement couplés, le demandeur du service ne connaît pas forcément le fournisseur.

Service Web (WebService)

Architecture :

Plusieurs architecture existent, mais les plus utilisés sont :

SOAP

REST

SOAP fonctionne avec une description de service fournis par WSDL

WSDL expose l'interface du service

REST fonctionne avec une description de service fournis par WADL ou RAML

WADL expose l'interface du service

RAML expose l'interface du service lié à des JSON

Exercice

Outil :

SoapUI

Site :

<http://www.oorsprong.com>

A faire :

Récupérer la liste des pays pour obtenir les informations

Pour aller plus loin : Trier par continent en amont