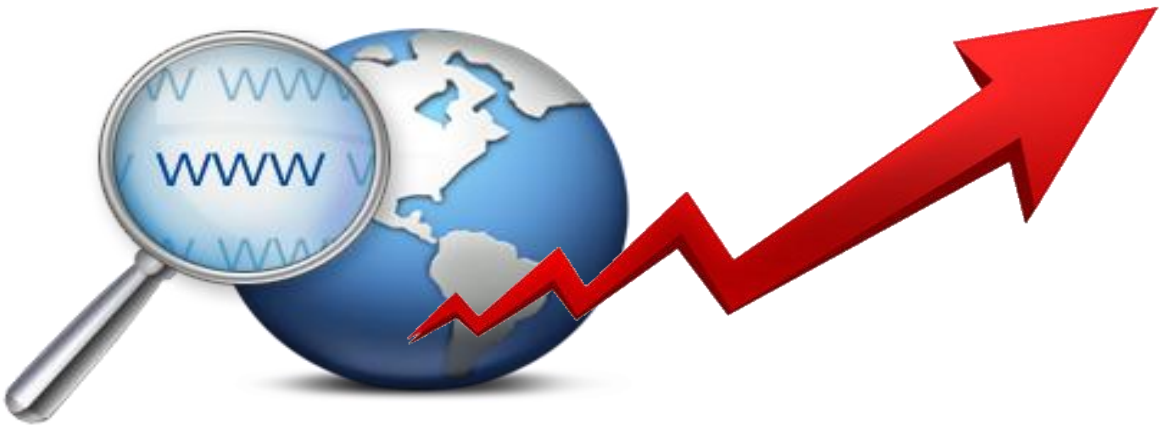
جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Université Abderrahmane Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

Module : Services Web (M2 – GL – ReSyD – IA (S3))



Cours 06 : REST

Présenté par Dr BRAHAMI EL BOUHISSI H.

2017-2018

Introduction

REST acronyme de **REpresentational State Transfert** : Concept introduit en 2000 dans la thèse de Roy FIELDING (un des créateurs du protocole HTTP).

REST est une manière de construire une application pour les systèmes distribués comme le World Wide Web.

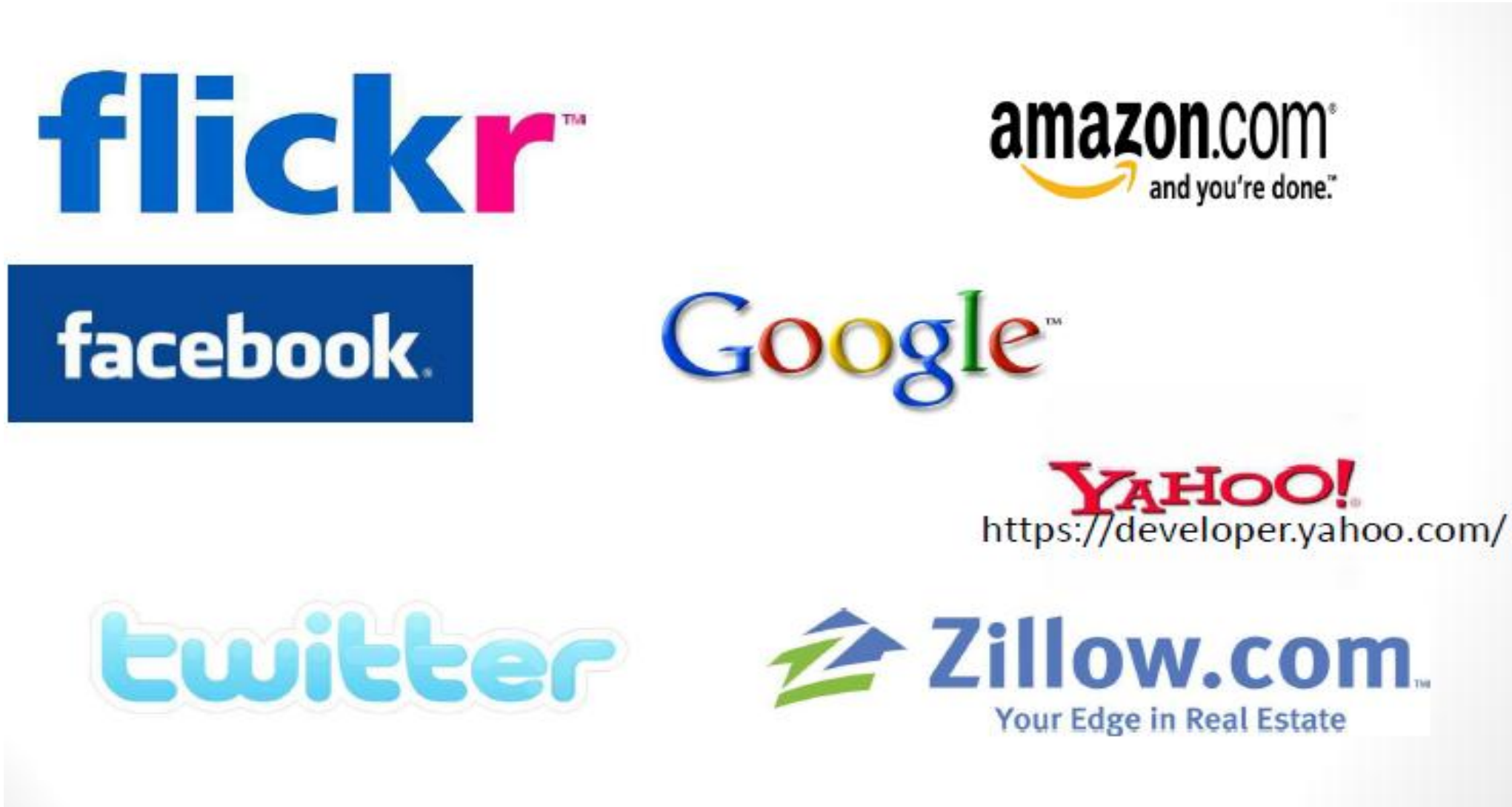
Ce n'est pas:

- Un format
- Un protocole
- Un standard
- Est un style d'architecture inspiré de l'architecture WEB

- Permet l'envoi de messages sans enveloppe SOAP et dans un encodage libre (XML, JSON, binaire, simple texte).

REST n'est pas un standard: Pas de recommandation du W3C

Services REST : Fournisseurs



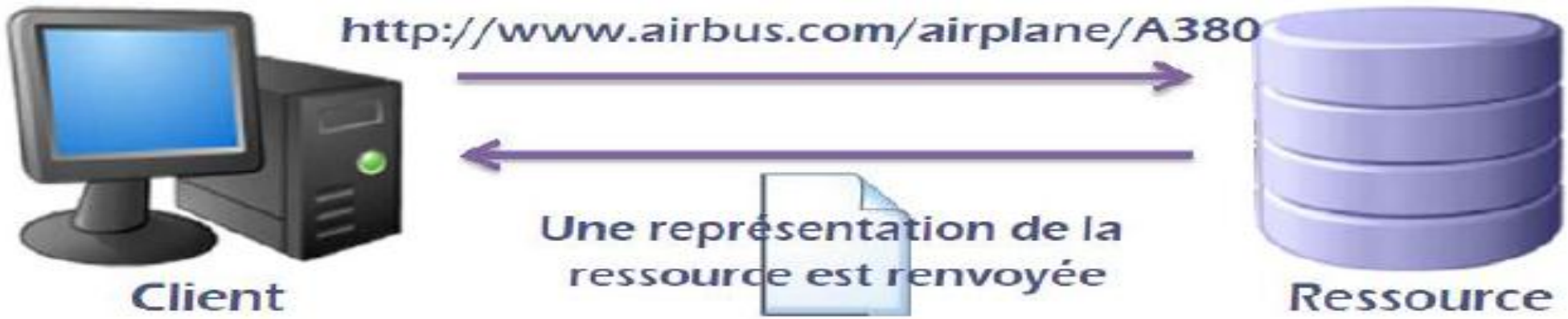
REST

REST est un style d'architecture qui repose sur le protocole HTTP : On accède à une ressource (par son URI unique) pour procéder à diverses opérations (GET lecture / POST écriture / PUT modification / DELETE suppression), opérations supportées nativement par HTTP.

Architecture Orientée Ressource

Toute information qui peut être nommée est une ressource

Une ressource est identifiée par un identificateur (URI)



REST : Exemple (1)

Supposons que nous voulons réaliser un serveur REST pour gérer les livres d'une bibliothèque. Nous devons pouvoir ajouter (*POST*), modifier (*PUT*), Lire (*GET*) et Supprimer (*DELETE*) ces livres (la ressource à manipuler).

L'adresse de notre bibliothèque représente le « point terminal » (endpoint) : http://bibliotheque/. (si notre bibliothèque n'était pas uniquement un serveur REST, notre point terminal pourrait être http://bibliotheque/rest/). Le point terminal n'est ni plus ni moins l'adresse de notre web service.

Notre bibliothèque contient des ressources, en particulier des livres, qui pourront être manipulés à une URI formée par convention de la sorte : **http://point_terminal/nom_de_ressource/,** soit dans notre exemple :

http://bibliotheque/livre/

REST : Exemple (2)

Nous pourrons effectuer plusieurs manipulation sur ces livres :

- Les lire : Requête de type *GET* sur **http://bibliotheque/livre/ID_DU_LIVRE_A_LIRE**
- En écrire : Requête de type *POST* sur **http://bibliotheque/livre**/. Le corps du message POST représente le contenu du nouveau livre à créer. A la charge de la bibliothèque d'affecter un identifiant à notre nouveau livre.
- Les modifier : Requête de type *PUT* sur **http://bibliotheque/livre/ID_DU_LIVRE**. Le corps du message PUT représente le contenu modifié du livre d'identifiant ID_DU_LIVRE.
- Les supprimer : Requête de type *DELETE* sur **http://bibliotheque/livre/ID_DU_LIVRE**.

REST : Format d'échange

REST n'impose pas de format d'échange entre client et serveur. Vous êtes libre de représenter vos données en XML, en JSON, en PHP sérialisé ou dans tout autre format.

Pour tester un service REST, nous pouvons utiliser notre navigateur WEB (au moins pour les requêtes de lecture).

Par exemple, pour tester les API REST de Flickr, les liens suivants sont parfaitement opérationnels :

- **Avec un format d'échange JSON :**

jsonFlickrApi({"stat":"fail","code":100,"message":"Invalid API Key (Key has invalid format)"})

- **Avec un format d'échange XML :**

<rsp stat="fail">

<err code="111" msg="Format "xml" not found"/>

</rsp>

Services REST : Exemples formats JSON et XML

GET https://www.googleapis.com/urlshortener/v1/url?shortUrl=http://goo.gl/fbsS

```
{
 "kind": "urlshortener#url",
 "id": "http://goo.gl/fbsS",
 "longUrl": "http://www.google.com/",
 "status": "OK"
}
```

Représentation des données en JSON

GET http://localhost:8080/librarycontentrestwebservice/contentbooks/string

```
<?xml version="1.0"?>
<details>
    Ce livre est une introduction sur la vie
</details>
```

Représentation des données en XML

Description de services Web REST

Langages : WSDL 2.0,WADL, RSDL (Restful Service Description Language), SERIN (Semantic RESTFul Interfaces).

WSDL2.0 : Évolution de *Web Service Description Language recommandé en 2007 par le W3C.* Il permet de spécifier un *binding HTTP au lieu de SOAP,*

WADL (Web Application Description Language) : Un langage de description XML de services de type REST.

Permet une description de services par éléments de type: ressource, méthode, paramètre, requête, réponse.

Pas assez de Framework qui supportent la description WADL

Description de services Web REST : WADL

WADL

ELEMENTS WADL	DESCRIPTION
<application>	La racine d'une description WADL
<ressources base=…….>	• Un conteneur pour les ressources que l'application fournit • L'attribut base définit l'URI pour les ressource
<resource id="" path="">	• Décrit la ressource que l'application fournit • L'attribut id identifie l'élément ressource • L'attribut path fournit une URI relative pour l 'identifiant de la ressource.
<method name=" " id="">	• Fils d'élément ressource ou application • L'attribut name définit les méthodes HTTP
<request> <response>	• Request: décrit une requête à la méthode HTTP sur une ressource • Response: décrit la sortie en réalisant la méthode HTTP sur le ressource

Description de services Web REST : WADL

EXEMPLE de description WADL pour l'application Yahoo News Search

```
<?xml version="1.0"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd"
 xmlns:tns="urn:yahoo:yn"   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:yn="urn:yahoo:yn"
 xmlns:ya="urn:yahoo:api"   xmlns="http://wadl.dev.java.net/2009/02">
  <grammars> <include  href="NewsSearchResponse.xsd"/> <include href="Error.xsd"/>
  </grammars>
  <resources base="http://api.search.yahoo.com/NewsSearchService/V1/">
    <resource path="newsSearch">
      <method name="GET" id="search">
        <request>
          <param name="appid" type="xsd:string" style="query" required="true"/>
          <param name="query" type="xsd:string"  style="query" required="true"/>
          <param name="type" style="query" default="all">
            <option value="all"/> <option value="any"/> <option value="phrase"/>
          </param>
          <param name="results" style="query" type="xsd:int" default="10"/>
          <param name="start" style="query" type="xsd:int" default="1"/>
          <param name="sort" style="query" default="rank">
            <option value="rank"/> <option value="date"/>
          </param>
          <param name="language" style="query" type="xsd:string"/>
        </request>
        <response status="200">
          <representation mediaType="application/xml" element="yn:ResultSet"/>
        </response>
        <response status="400">
          <representation mediaType="application/xml" element="ya:Error"/>
        </response>
      </method>
    </resource>
  </resources>
</application>
```

Description des espaces de noms

Description de Grammaire XML utilisée par le serv

Description des ressources Et méthodes HTTP utilisées

Plus facile à comprendre, à interpréter et à écrire qu'un WSDL

Quelles sont les différences entre SOAP et REST ?

REST et SOAP sont tous les deux des architectures utilisées pour fournir des services web. Contrairement à ce que l'acronyme SOAP laisse entendre (Simple Object Access Protocol),

REST est souvent utilisé lorsque la simplicité de mise en œuvre est recherchée.
REST est lisible (pas d'enveloppe XML superflue) et facile à tester (un navigateur suffit) tout en étant facile de mise en œuvre (un script PHP classique peut souvent être considéré comme RESTful).

SOAP reste toutefois intégré dans de nombreux outils de développements (possibilité d'export de classes en web services, possibilité de génération automatique de clients à partir des WSDL) et permet des contrôles forts sur les types de données attendus.

	REST	SOAP
Standardisé	Non	Oui
Sécurité	Non (restreinte par l'emploi des méthodes HTTP)	Oui (WS-Security)
Complexité	Non	Oui (lourdeur)

Un exemple de web service REST (1)

http://api.geonames.org/findNearestAddressJSON?lat=40.771&lng=-73.97&username=demo

Ce web service retrouve la rue et l'adresse la plus proche pour une paire **lng / lat** donnée (pour notre exemple les coordonnées désignent Central Park à New York : lat=40.771 lng=-73.97).

L'Url du web service est : http://api.geonames.org/findNearestAddress?

Les Paramètres sont: **lat**, **lng**, (désignant la latitude et la longitude)

Ce web service est disponible uniquement pour les Etats-Unis

Un exemple de web service REST (2)

Résultat dans le navigateur au format JSON: