

# Architecture Orientée Service, JSON et API REST

Ludovic DENOYER - ludovic.denoyer@lip6.fr

UPMC

27 janvier 2016

## Précédemment, en LI328

- Architecture générale du projet
- Programmation serveur
- Servlet/TOMCAT

## Aujourd'hui

- Quelques mots sur les SOA
- API - REST
- Le format JSON
- API - REST et Servlet
- API - SOAP (un peu)

Différents paradigmes :

- Procédures
- Programmation Orientée Objet
- Programmation Orientée Composants
- Programmation Orientée Service

Différents paradigmes :

- Procédures

(source wikipedia)

La **programmation procédurale** est un paradigme de programmation basé sur le concept d'appel procédural. Une procédure contient simplement une série d'étapes à réaliser. N'importe quelle procédure peut être appelée à n'importe quelle étape de l'exécution du programme.

- Programmation Orientée Objet
- Programmation Orientée Composants
- Programmation Orientée Service

# Paradigmes de programmation

Différents paradigmes :

- Procédures
- Programmation Orientée Objet

(source wikipedia)

Un objet représente un concept, une idée ou toute entité du monde physique. Il possède une structure interne et un comportement, et il sait communiquer avec ses pairs. Il s'agit donc de représenter ces objets et leurs relations ; la communication entre les objets via leurs relations permet de réaliser les fonctionnalités attendues, de résoudre le ou les problèmes.

- Programmation Orientée Composants
- Programmation Orientée Service

# Paradigmes de programmation

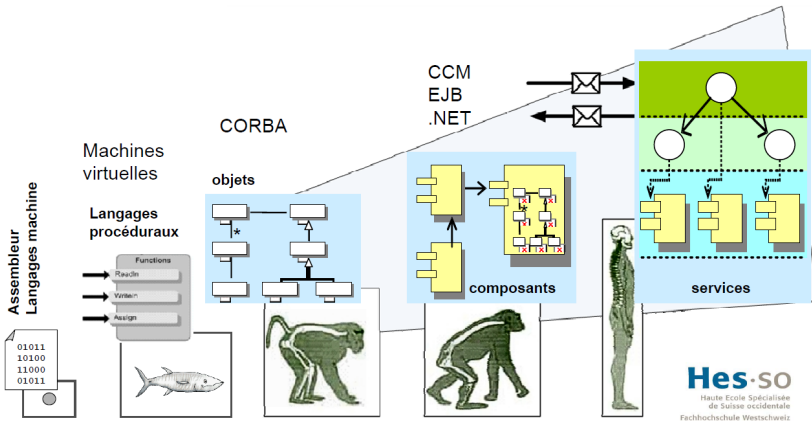
Différents paradigmes :

- Procédures
- Programmation Orientée Objet
- Programmation Orientée Composants

(source wikipedia)

La programmation orientée composant (POC) consiste à utiliser une approche modulaire au niveau de l'architecture d'un projet informatique, ce qui permet d'assurer au logiciel une meilleure lisibilité et une meilleure maintenance. Les développeurs, au lieu de créer un exécutable monolithique, se servent de briques réutilisables.

- Programmation Orientée Service



➤ *Niveaux d'abstraction grandissant*

## Architecture orientée Service

Une architecture orientée services (notée SOA pour Services Oriented Architecture) est une architecture logicielle s'appuyant sur un ensemble de services simples. Elle permet de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées services, fournies par des composants et de décrire finement le schéma d'interaction entre ces services.

Lorsque l'architecture SOA s'appuie sur des web services, on parle alors de WSOA, pour **Web Services Oriented Architecture**.



## Avantages des WS

- Les services Web fournissent l'interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes.
- Les services Web utilisent des standards et protocoles ouverts.
- Les protocoles et les formats de données sont au format texte dans la mesure du possible, facilitant ainsi la compréhension du fonctionnement global des échanges.
- Basés sur le protocole HTTP, les services Web peuvent fonctionner au travers de nombreux pare-feu sans nécessiter des changements sur les règles de filtrage.
- Les outils de développement, s'appuyant sur ces standards, permettent la création automatique de programmes utilisant les services Web existants.

## Inconvénients des WS

- Les normes de services Web dans certains domaines sont actuellement récentes.
- Les services Web souffrent de performances faibles comparée à d'autres approches de l'informatique répartie telles que le RMI, CORBA, ou DCOM.[réf. nécessaire]
- Par l'utilisation du protocole HTTP, les services Web peuvent contourner les mesures de sécurité mises en place au travers des pare-feu.

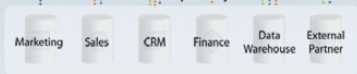
## Before SOA

Siloed · Closed · Monolithic · Brittle

### Application Dependent Business Functions



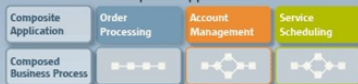
### Data Repository



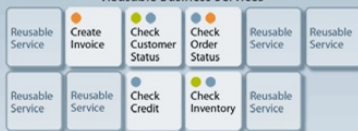
## After SOA

Shared services · Collaborative · Interoperable · Integrated

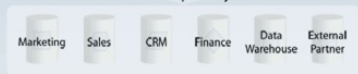
### Composite Applications



### Reusable Business Services



### Data Repository



# Les Web Services

- Un Service est Autonome  
(et sans état)



- Les Frontières entre services  
sont Explicites



- Un Service expose un Contrat



Conditions Générales de Vente  
Règlement Intérieur  
Vos droits/Vos devoirs

- Les services communiquent par  
messages



Source A. Occeila

## REST

- REST = REpresentational State Transfer
- C'est une manière de construire une application pour les systèmes distribués
- REST n'est pas un protocole ou un format, c'est un style d'architecture
- Il est de plus en plus utilisé pour la réalisation d'architectures orientées services utilisant des services Web destinés à la communication entre machines.

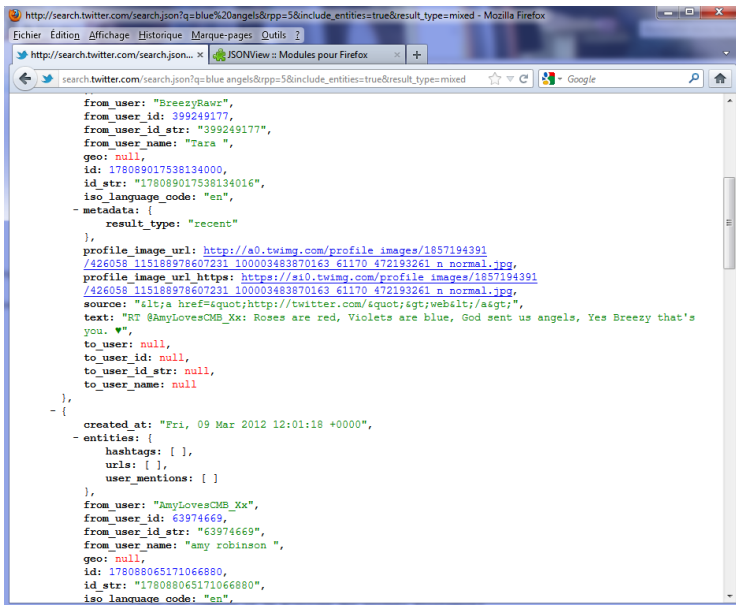
## Principes

- l'URI est important : connaître l'URI doit suffire pour nommer et identifier une ressource.
- HTTP fournit toutes les opérations nécessaires (GET, POST, PUT et DELETE, essentiellement).
- Chaque opération est auto-suffisante : il n'y a pas d'état.
- Utilisation des standards hypermedia : HTML ou XML ou

## JSON

*Référence* : RESTful Web Services, par Leonard Richardson et Sam Ruby

# API REST : Exemple Twitter



```
http://search.twitter.com/search.json?q=blue%20angels&rpp=5&include_entities=true&result_type=mixed - Mozilla Firefox
Fichier Edition Affichage Historique Marque-pages Outils ?
http://search.twitter.com/search.json... x JSONView :: Modules pour Firefox x +
search.twitter.com/search.json?q=blue%20angels&rpp=5&include_entities=true&result_type=mixed Google
{
  "from_user": "BreezyRawr",
  "from_user_id": 399249177,
  "from_user_id_str": "399249177",
  "from_user_name": "Tara ",
  "geo": null,
  "id": 178089017538134000,
  "id_str": "178089017538134016",
  "iso_language_code": "en",
  "metadata": {
    "result_type": "recent"
  },
  "profile_image_url": "http://a0.twimg.com/profile_images/1857194391/426058_115188978607231_100003483870163_61170_472193261_n_normal.jpg",
  "profile_image_url_https": "https://s0.twimg.com/profile_images/1857194391/426058_115188978607231_100003483870163_61170_472193261_n_normal.jpg",
  "source": "<a href='\"http://twitter.com/\"'>web</a>",
  "text": "RT @AmyLovesCMB_Xx: Roses are red, Violets are blue, God sent us angels, Yes Breezy that's you. ♥",
  "to_user": null,
  "to_user_id": null,
  "to_user_id_str": null,
  "to_user_name": null
},
- {
  "created_at": "Fri, 09 Mar 2012 12:01:18 +0000",
  "entities": {
    "hashtags": [ ],
    "urls": [ ],
    "user_mentions": [ ]
  },
  "from_user": "AmyLovesCMB_Xx",
  "from_user_id": 63974669,
  "from_user_id_str": "63974669",
  "from_user_name": "amy robinson ",
  "geo": null,
  "id": 178088065171066880,
  "id_str": "178088065171066880",
  "iso_language_code": "en",
```

# API REST : Exemple Flickr

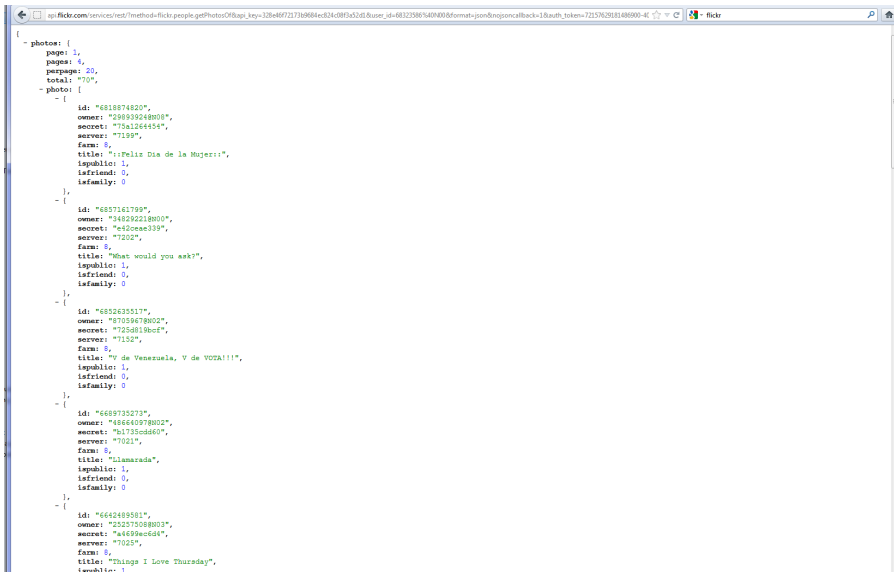
api.flickr.com/services/rest/?method=flickr.people.getPhotosOf&api\_key=32b86f7217316964ec324c0f3a52d1&user\_id=6832598%40N00&format=rest&auth\_token=72157629181486900-40a224e69e843646a4

Aucune information de style ne semble associée à ce fichier XML. L'arbre du document est affiché ci-dessous.

```
<?xml version="1.0"?>
<rsp stat="ok">
  <photos page="1" pages="4" perpage="20" total="70" has_next_page="1">
    <photo id="6818874820" owner="29893924@N08" secret="75a1264454" server="7199" farm="8" title="Feliz Dia de la Mujer." ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6857161799" owner="34829221@N00" secret="e42ceac339" server="7202" farm="8" title="What would you ask?" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6852635517" owner="8705967@N02" secret="725d819bec" server="7152" farm="8" title="V de Venezuela, V de VOTA!!!" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6689735273" owner="48664097@N02" secret="b1735cdd60" server="7021" farm="8" title="Llamarada" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6642489581" owner="25257508@N03" secret="a4699ec6d4" server="7025" farm="8" title="Things I Love Thursday" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6641187815" owner="48739681@N05" secret="9c5144ca5a" server="7016" farm="8" title="~ Feliz Nit de Reis ~ Feliz Noche de Reyes ~" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6596007895" owner="34829221@N00" secret="961ed2d366" server="7165" farm="8" title="2011 in Review" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6593866295" owner="38838924@N03" secret="f867c83f24" server="7153" farm="8" title="Mi 2011 en flickr" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6590116673" owner="8705967@N02" secret="23bd13d66b" server="7018" farm="8" title="Navidad" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6587696631" owner="8332135@N03" secret="63ce7bd96a" server="7142" farm="8" title="The sky is the limit..." ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6581955027" owner="63307805@N00" secret="cfabed377" server="7007" farm="8" title="My year in pictures" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6581955027" owner="39307146@N08" secret="7b56b63043" server="7171" farm="8" title="My year in pictures" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="656645059" owner="23523125@N08" secret="4c73a0e85" server="7147" farm="8" title="Feliz Navidad..." ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6564959719" owner="29219049@N06" secret="420005be9d" server="7153" farm="8" title="Feliz Navidad!!!" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6560620287" owner="40654531@N05" secret="8f0dc0970a" server="7001" farm="8" title="mi navidad, tu navidad? Explored Dec 23, 2011 #6 Muchas Gracias!" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6553379789" owner="49762065@N08" secret="20b628586" server="7002" farm="8" title="." ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6538048271" owner="37763325@N08" secret="66c295f74" server="7171" farm="8" title="Felices Fiestas! ~ Happy Holidays!" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6526003925" owner="66881369@N06" secret="ebb8093294" server="7020" farm="8" title="Family lunch" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6475299671" owner="43616712@N03" secret="aa8110b0f7" server="7146" farm="8" title="lovely support" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6358869783" owner="25257508@N03" secret="205188c38" server="6019" farm="7" title="Friday Favorites" ispublic="1" isfriend="0" isfamily="0"/>
  </photos>
</rsp>
```



# API REST : Exemple Flickr



```
[
  - photos: {
    page: 1,
    pages: 4,
    perpage: 20,
    total: "70",
    - photo: [
      - {
        id: "6818974820",
        owner: "298939248N08",
        secret: "75a1264454",
        server: "7199",
        farm: 8,
        title: "::Feliz Dia de la Mujer::",
        ispublic: 1,
        isfriend: 0,
        isfamily: 0
      },
    ],
    - {
      id: "6857161799",
      owner: "348292218N00",
      secret: "e42ceae339",
      server: "7202",
      farm: 8,
      title: "What would you ask?",
      ispublic: 1,
      isfriend: 0,
      isfamily: 0
    },
    ],
    - {
      id: "6852635517",
      owner: "87059678N02",
      secret: "725d819bcf",
      server: "7152",
      farm: 8,
      title: "V de Venezuela, V de VOTA!!!",
      ispublic: 1,
      isfriend: 0,
      isfamily: 0
    },
    ],
    - {
      id: "6689735273",
      owner: "486640978N02",
      secret: "b1735cd60",
      server: "7021",
      farm: 8,
      title: "Llamarada",
      ispublic: 1,
      isfriend: 0,
      isfamily: 0
    },
    ],
    - {
      id: "6642489581",
      owner: "252575088N03",
      secret: "a4699e6d4",
      server: "7025",
      farm: 8,
      title: "Things I Love Thursday",
      ispublic: 1
    }
  ]
}
```

# API REST : Exemple Flickr



# API REST : Exemple Flickr

Returns the comments for a photo

## Authentification

Cette méthode n'exige pas d'authentification.

## Arguments

**api\_key** (Obligatoire)

Your API application key. [See here](#) for more details.

**photo\_id** (Obligatoire)

The id of the photo to fetch comments for.

**min\_comment\_date** (Facultatif)

Minimum date that a comment was added. The date should be in the form of a unix timestamp.

**max\_comment\_date** (Facultatif)

Maximum date that a comment was added. The date should be in the form of a unix timestamp.

## Exemple de réponse

```
<comments photo_id="109722179">
  <comment id="6065-109722179-72057594077818641" author="35468159852@N01" authorname="Rev Dan Catt"
/>
</comments>
```

## Codes d'erreur

**1: Photo not found**

The photo id was either invalid or was for a photo not viewable by the calling user.

**100: Invalid API Key**

The API key passed was not valid or has expired.

**105: Service currently unavailable**

The requested service is temporarily unavailable.

**111: Format "xxx" not found**

The requested response format was not found.

**112: Method "xxx" not found**

The requested method was not found.

**114: Invalid SOAP envelope**

## Principes Généraux

- Les paramètres sont passés (à travers HTTP) en utilisant le protocole GET (le plus souvent)
- La réponse est un document sous forme JSON ou XML

## Avantages

- Simplicité
- Lisibilité par l'humain
- Evolutivité
- Repose sur les principes du Web
- Représentations multiples

## Avantages

- Simplicité
- Lisibilité par l'humain
- Evolutivité
- Repose sur les principes du Web
- Représentations multiples

## Inconvénients

- Sécurité restreinte par l'emploi de HTTP
- Cible uniquement l'appel de ressources
  - Architecture orientée ressources (ROA)
  - ou Architecture orientée données (DOA)

Les Services Web étendus (SOAP) et les Services Web REST sont différents par le fait que :

- Services Web étendus (SOAP)
- Avantages
  - Standardisé
  - Interopérabilité
  - Sécurité (WS-Security)
  - Outillé
- Inconvénients
  - Performances (enveloppe SOAP supplémentaire)
  - Complexité, lourdeur
  - Cible l'appel de service

- JavaScript Object Notation
- Initialement créé pour la sérialisation et l'échange d'objets JavaScript
- Langage pour l'échange de données semi-structurées (et éventuellement structurées)
- Format texte indépendant du langage de programmation utilisé pour le manipuler.
- Assez proche du XML, mais moins c\*\*\*\* embêtant

Un document JSON ne comprend que deux éléments structurels :

- des ensembles de paires nom / valeur ;
- des listes ordonnées de valeurs.

Ces mêmes éléments représentent 3 types de données :

- des objets ;
- des tableaux ;
- des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou null.



```
{
  "menu":
  {
    "id": "file",
    "value": "File",
    "popup":
    {
      "menuitem":
      [
        { "value": "New", "onclick": "CreateNewDoc()" },
        { "value": "Open", "onclick": "OpenDoc()" },
        { "value": "Close", "onclick": "CloseDoc()" }
      ]
    }
  }
}
```

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

```
{
  person: {name: "alan", phone: 3127786, email: "agg@abc.com"},
  person: &314
    {name: {first: "Sara", last: "Smith-Green"},
      phone: 2136877,
      email: "sara@math.xyz.edu",
      spouse: &443},
  person: &443
    { name: "Fred Green",
      phone: 7786312,
      Height: 183,
      spouse: &314 }
}
```

Nous allons utiliser la librairie JSONObject de JAVA.

- Un **JSONObject** est une collection non-ordonnée de paires nom/valeur
- La méthode **put** permet d'ajouter une paire à l'objet
- Le texte de la méthode **toString** est conforme à la spécification JSON

```
1 myString = new JSONObject().put("JSON", "Hello,World!").toString();
2 // myString is {"JSON": "Hello, World"}
```

# TOMCAT et REST

```
1
2 public class Example extends HttpServlet
3 {
4     ....
5
6     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException
7     {
8         response.setContentType("text/plain");
9         String prenom=request.getParameter("prenom");
10        if (prenom==null)
11        {
12            JSONObject json=new JSONObject();
13            try {
14                json.put("error","Missing parameter");
15            } catch (JSONException e) {
16                e.printStackTrace();
17            }
18            response.getWriter().println(json.toString());
19        }
20        else
21        {
22            JSONObject json=new JSONObject();
23            try {
24                json.put("output","OK");
25            } catch (JSONException e) {
26                // TODO Auto-generated catch block
27                e.printStackTrace();
28            }
29            response.getWriter().println(json.toString());
30        }
31    }
32 }
```

## Conclusion

Vous connaissez :

- TOMCAT
- REST

## TD / TP

- Spécifier les services à développer
- Mettre en place l'architecture des services (en attendant l'accès au BDs)

Prochain Cours : SQL et JDBC

# Quelques mots sur le projet

- Vous devez tenir à jour un fichier spécifiant les services développés (à développer)
- Les rappels de cours seront évalués lors des TDs
- Vous devez séparer les traitements des servlets et tester vos traitements avant de déployer les servlets. Une (grosse) erreur dans vos servlet risque de faire planter tout le serveur et donc tous les servlets de vos camarades
- Les binomes de travail devront être fixés **définitivement** lors du prochain TD