

3I017 - Technologies du Web

Laure Soulier - laure.soulier@lip6.fr

Sorbonne Université

21 janvier 2019

Organisation de l'UE

Acquisition de techniques pour le développement de sites Web "modernes"

- Architectures des Sites "sociaux"
- Développement de services Web
- Développement de d'interfaces homme/machine
- Traitement de grandes masses de données

Un enseignement "concret" basé sur la manipulation de technologies

- Cours : Présentation des différentes technologies et de leur articulation
- TD : Prise en mains des technologies
- TP : Mise en oeuvre de ces technologies

Les TP sont tous structurés autour du développement d'un site de type
Twitter

Objectifs de l'UE

Développement d'un site type réseau social

The screenshot shows a web interface for a social network, styled after Twitter. At the top is a dark navigation bar with the 'twitter' logo, a search bar labeled 'Rechercher', and links for 'Accueil', 'Profil', 'Messages', and 'Suggestions'. A user profile icon for 'tsinapah_sly' is on the right. Below the navigation bar, the main content area is divided into two columns. The left column, titled 'Quoi de neuf ?', contains a tweet composition box with a 'Tweeter' button showing '139' retweets, and a 'Fil' tab. Below the tab are three tweets: one from 'misspress' about Raphaël Enthoven, one from 'jokerwoman' about a population study, and one from '20minutes' about a job opening. The right column contains sections for 'Vos Tweets 1', 'Abonnements 2', a list of steps to get started (like 'Recevez Twitter sur votre téléphone'), and a 'Trouver des amis' section.

twitter Rechercher Accueil Profil Messages Suggestions tsinapah_sly

Quoi de neuf ?

139 **Tweeter**

Fil @tsinapah_sly Activité Recherches Listes

misspress Melissa Bounoua
Pourquoi Raphaël Enthoven en veut-il autant à @quentingirard ?
[huffingtonpost.fr/raphael-enthov...](#)
Il y a 1 minute

jokerwoman Anastasia Levy
Pour la première fois depuis le gouvernement de Vichy, la France prépare un fichier total de sa population [bit.ly/AEfHrl](#) sur @owni
Il y a 45 minutes

20minutes 20minutes.fr
On recrute un reporter rédacteur web (Pôle entertainment : people, médias, télé, cinéma / CDI) - LM et CV à [recrutement@20minutes.fr](#)
Il y a 16 heures
Retweeté par misspress

Vos Tweets 1
22 Sept 10 : rien

Abonnements 2

Trouvez des comptes à suivre : [Par catégories](#) · [Trouver vos amis](#)

Et ensuite ? · [masquer les prochaines étapes](#)

1. **Recevez Twitter sur votre téléphone**
 - Configurer les notifications sur mobile
 - Télécharger une application Twitter sur votre téléphone
2. **Configurez votre profil**
 - Charger une image de profil
 - Rédigez une courte biographie

Trouver des amis
Utilisez les services ci-dessous pour trouver vos connaissances sur Twitter.

Evaluation de l'UE

- L'UE n'est pas une UE difficile, mais **une UE dense** : une à deux technologies par semaine, aucun retour en arrière
- 50% : contrôle continu
 - ▶ 3 contrôle de TD (3pts)
 - ▶ Evaluation serveur à mi-parcours (2pts)
 - ▶ Soutenance de projet (15pts)
- 50% : Examen terminal sur feuille. L'examen est long et difficile.....

Evaluation en TD

- 3 interrogations (indicatif : TME 2, 4 et 8)
- Possibilité d'être interrogés sur ce qui a été vu en amphi et non vu en TD/TME → **Relire le cours avant d'aller en TD/TME**

Evaluation serveur à mi-parcours

- Se déroule en TME 5
- Faire le point sur ce qui a été fait et ce qui manque
- Permet de commencer le développement du client la semaine suivante

Soutenance de projet

- Projet en binôme ou monôme
- Réalisation de Twister côté Client + côté Serveur
- Implémentation de fonctionnalités **obligatoires** - voir TD/TME 3
- Implémentation de fonctionnalités **additionnelles**
- Présentation de la réalisation (TME 11) + Modifications à apporter au projet
 - ▶ Les modifications sont la "clefs" de l'évaluation

A réaliser :

- Page d'accueil
- Page de profil
- Formulaires de connexion / enregistrement
- Fonctions d'ajout de contacts
- Fonctionnalités permettant de poster un commentaire
- Fonctions de recherches thématiques
- Statistiques (centres d'intérêts, amis les plus actifs, taux de réponse, etc...)
- Plus tout autre fonctionnalité originale....

Planning de l'UE

S	Cours	TD	TME
1	Intro - AOS/Services	-	-
2	Modélisation (TD)	AOS (Jeu)	Prise en main
3	BD rel (MySQL)	Services (I)	Services
4	Big Data/NoSQL - MongoDB	MySQL	MySQL
5	HTML/CSS/Javascript	MongoDB	MongDB (I)
6	JQuery/ Communication C-S	Eval serveur (4h TME)**	
7	Cloud Computing	HTML/CSS (4h TME)	
8	Référencement	Javascript	Dev client
9	Pas cours	Jquery (I)	Dev Client
10	Moteur de recherche - MapReduce	Ajax	Dev client
11	Annales	MapReduce	MapReduce
12	-	Soutenance projet (2h TME)**	

(I) : interro TD

** : évaluation TME mi-parcours et projet

Concrètement

Attention :

On a que 11 semaines \Rightarrow Il faut utiliser les technos récentes et robustes.

Choix de l'enseignement

On a décidé de s'appuyer sur vos connaissances acquises.

- Client Web (et non pas client mobile de type iPhone/Android)
- Serveur Tomcat (JAVA)
- On garde une base SQL (mais on rajoute du NoSQL)

Ce que l'on va utiliser :

- Présentation graphique
 - HTML, CSS, Javascript
- Serveur de service Web
 - Apache Tomcat
- Language développement côté serveur
 - JAVA
- Communication client - serveur
 - AJAX avec format JSON
- Bases de Données
 - MySQL, MongoDB

En résumé...

Objectif double du cours

- ➊ Vous enseigner - donner un point d'entrée - pour un ensemble de technologies
- ➋ Vous "éclairer" sur les évolutions actuelles de l'informatique

→ Si le point 1 n'est qu'affaire de "débrouillardise", le point 2 est celui qui doit rester dans un coin de votre tête.

→ Si les technos doivent être maîtrisées, le plus important demeure la compréhension de leur "imbrication"

→ Les différents points abordés recoupent des notions étudiées en profondeur en Master : programmation distribuée (Master SAR), réseau (Master RES), développement logiciel (Master STL), traitement de données (Master DAC)

Les TPs sont tous structurés autour du développement d'un site de type Twitter

Cela implique :

- Les TPs sont additifs \Rightarrow retard/absence à un TP doit être rattrapé avant le TP suivant
- Les TDs introduisent les TPs \Rightarrow absence en TD = grosses difficultés en TP
- Les cours présentent les technologies \Rightarrow absence en cours = retard en TP

En résumé...

Critères de réussite de l'UE Côté étudiant :

- Les étudiants ont fait preuve d'autonomie
- Ce cours permet d'acquérir un socle pour découvrir " par soi-même"
- Les étudiants ont acquis une compétence professionnelle
- Vous êtes capables de monter votre Twitter/Facebook/Megaupload vous même dès maintenant

Répétition : Cette UE est structurée autour d'un projet qui doit être la source de motivation de chacun. Les enseignants seront ouverts (et favorables) à toute proposition/personnalisation de l'UE. Pas de projet/implication \Rightarrow pas de "diplôme".

Pour vous aider : Forum de discussion

- Utilisation d'une messagerie pour les discussions entre étudiants + enseignants/étudiants
- Vous pouvez poser toutes les questions...
- Vous répondre entre vous (on répondra également)
- MAIS ON NE PARTAGE PAS DE CODE (Je sais tout!!!!)
- Inscription : https://channel.lip6.fr/signup_user_complete/?id=3agxqa7bktg5xxtgwcmrb3pyiw

Amphi

TD

Projet

Logistique - Examens

Chargés de cours/TD/TME

Random - ActuWeb

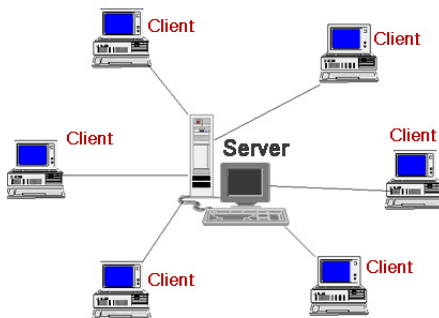
La team TechnoWeb !

- Laure Soulier : responsable d'UE, chargée du cours, TD/TME
- Thomas G rald : charg  TD/TME
- Yifu Chen : charg  TD/TME
- Marwan Ghanem : charg  TD/TME

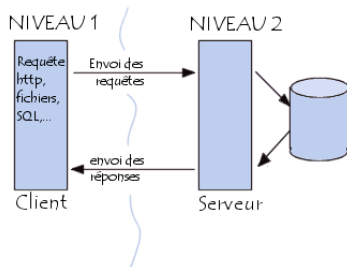
Questions ?

Architecture Client-Serveur

Architecture Client-Serveur

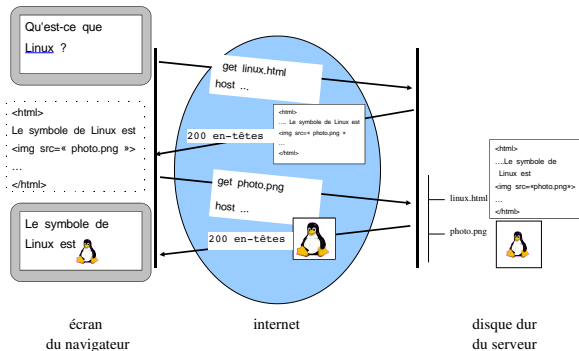


Architecture Client server



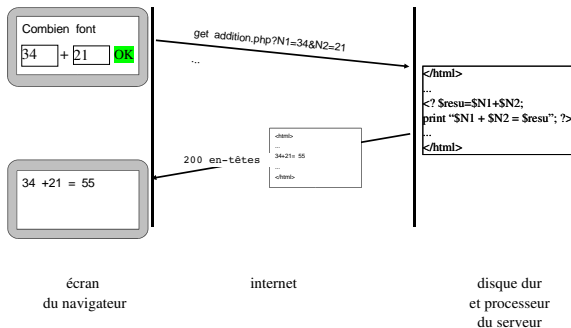
- Client (HTML, CSS, JAVASCRIPT, ...)
 - ▶ Il établit la connexion au serveur à destination d'un ou plusieurs ports réseaux
 - ▶ lorsque la connexion est acceptée par le serveur, il communique /interroge le serveur.
- Serveur
 - ▶ Il attend une connexion entrante sur un ou plusieurs ports
 - ▶ A la connexion d'un client sur le port, il communique avec le client.

Pages Web stockées sur un serveur Web

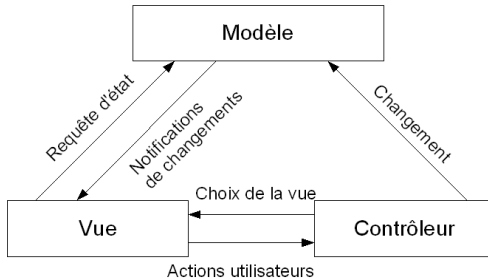


Développement d'un site type réseau social

Mise en place de services Web



Modèle - Vue - Contrôleur (MVC)



Serveurs PHP vs. JAVA

- Deux grandes écoles

PHP / Symfony / Doctrine....

- Avantages

- ▶ Apprentissage facile
- ▶ Multi-plateformes
- ▶ Déploiement d'application simples
- ▶ Nécessite peu de configuration

- Inconvénients

- ▶ Pas de typage (langage de script) → maintenance et debug très compliqués
- ▶ Pas de rétro-compatibilité (pas adaptable si codé avec l'ancienne version de PHP)

JAVA / TOMCAT / Servlets

- Avantages

- ▶ Plus adapté pour des applications web qui nécessitent des précisions dans les opérations
- ▶ Etape de compilation qui facilite la maintenance
- ▶ Eco système plus développé que PHP, rétro-compatibilité

- Inconvénients

- ▶ Système lourd lors du déploiement/config
- ▶ Apprentissage plus compliqué que PHP

De très nombreuses technos

- Langages développement Web
 - Java, PHP, ASP, etc...
- Frameworks de développement
 - Struts, Java Server Faces, Flex, Open Lazlo, etc...
- Serveurs de services Web
 - Tomcat, IIS, Google Web Server, etc...
- Moteurs de bases de données
 - MySQL, Hadoop, MongoDB, etc...
- Formats d'échange
 - HTML, XML, JSON, SOAP, etc...

Quelles Technologies ?

Technologies Client

- **HTML**
- **CSS**
- **JavaScript**
- **AJAX**

Technologies d'échange de données

- **JSON**
- **XML**
- **API/REST - Web API**

Quelles Technologies ?

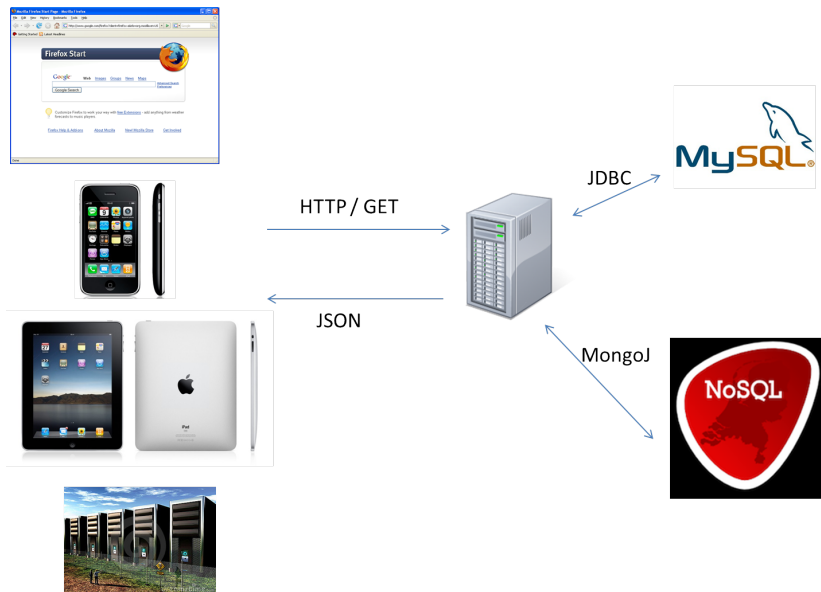
Technologies Serveur

- Serveur Web (ici : TOMCAT)
- Base de données :
 - ▶ SQL (ici : MySQL)
 - ▶ NoSQL (ici : MongoDB)

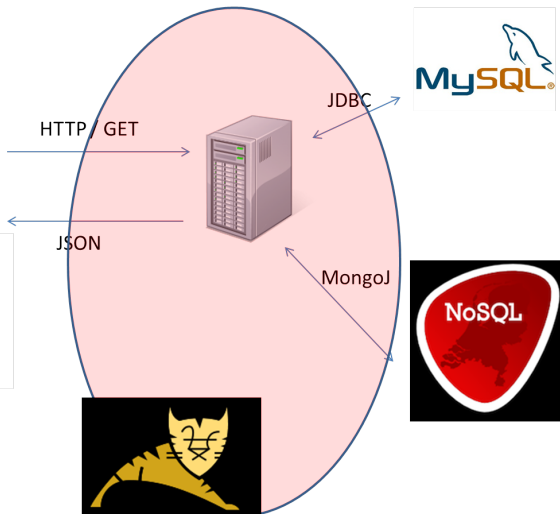
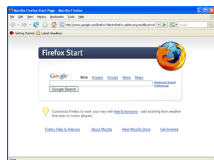
Technologies de traitement

- Map/Reduce

Web API



Web API et Tomcat



Concrètement

Etude de Cas

Le cours est structuré autour du développement "from scratch" d'un site Web de type Twitter incluant :

- Une interface Web pour les utilisateurs
- Une API disponible pour le développement d'applications
- Un serveur permettant le stockage de **grandes masses de données dynamiques**
- Une interface de traitement de données

Vers une Architecture Orientée Service

Paradigmes de programmation

Différents paradigmes :

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

Paradigmes de programmation : Procédures

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 : Procédures

```
int somme(int a, int b){  
    int somme=a+b;  
    .....  
    .....  
    .....  
    if (somme>0){  
        return somme;  
    }else{  
        return -10;  
    }  
}
```

Limites :

- Difficulté de réutilisation du code
- Lisibilité
- Maintenance

Paradigmes de programmation : programmation orientée objet

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 : programmation orientée objet

- Objet :
 - ▶ Données (variables d'instances)
 - ▶ Traitements (méthodes)
 - ▶ Principe d'encapsulation : on accède aux variables au travers des méthodes (accesseurs)
- Relations entre les objets :
 - ▶ Inclusion d'objets

```
1      public class Temperature{
2          public int tempe;
3          public int getTempe(){ return tempe; }
4      }
5
6      public class Radiateur {
7          public String nom;
8          public Temperature temp;
9      }
```

- ▶ Relations d'héritage

```
1      public class Vehicule{...}
2      public class Avion implements Vehicule{...}
3      public class Voiture implements Vehicule{...}
4      public class Camion implements Vehicule{...}
```

Paradigmes de programmation : Programmation Orientée Composants

Différents paradigmes :

- Procédures
- Programmation Orientée Objet
- Programmation Orientée Composants

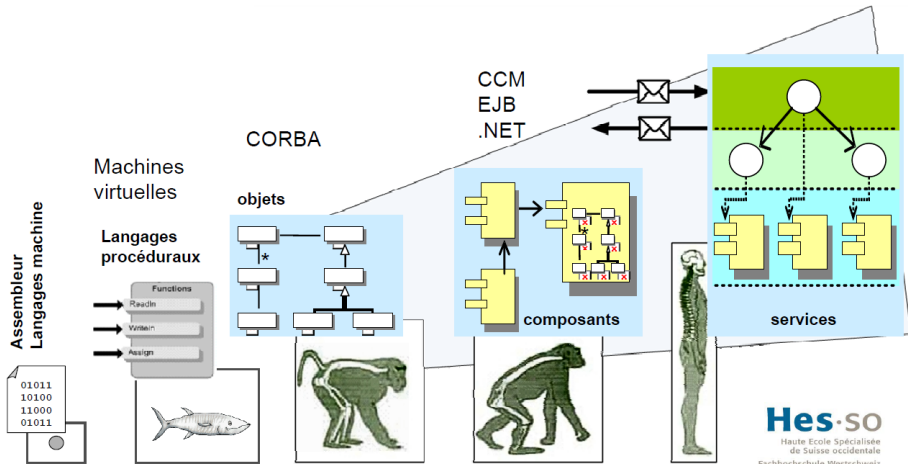
(source wikipedia)

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 (1 seul bloc), se servent de briques réutilisables.

- Programmation Orientée Service

Paradigmes de programmation : Programmation Orientée Composants

- Composant :
 - ▶ Code compilé, versionné et réutilisable : "module logiciel"
 - ▶ Composants isolés : pas besoin de connaître les modules dépendants pour les utiliser
 - ▶ Communication avec le monde extérieur avec port offert (export) et port requis (import)
- Avantages :
 - ▶ Sous-traitance / spécialisation
 - ▶ Facilité de mise à jour
 - ▶ Facilité de livraison ou déploiement
 - ▶ Langages différents entre les composants
- Inconvénients
 - ▶ Développement à long terme
 - ▶ Importance de la modélisation
 - ▶ Aucun contrôle total sur le projet
 - ▶ Propagation d'erreur quand les composants sont imbriqués



➤ *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.

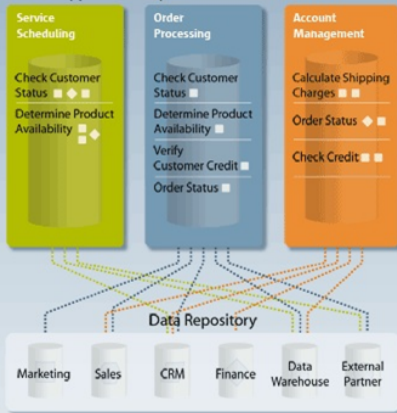
Inconvénients des WS

- 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

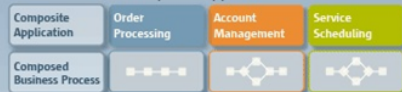
Application Dependent Business Functions



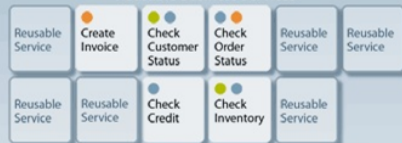
After SOA

Shared services · Collaborative · Interoperable · Integrated

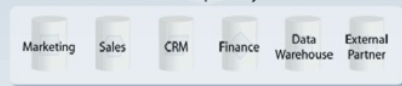
Composite Applications



Reusable Business Services



Data Repository



Les Web Services

- Un Service est Autonome (et sans état)



- Un Service expose un Contrat



Conditions Générales de Vente
Règlement Intérieur
Vos droits/Vos devoirs

- Les Frontières entre services sont Explicites



- Les services communiquent par messages



Source A. Occhetto

Tomcat et servlets

Tomcat

Apache Tomcat est...

- un conteneur libre de servlets et JSP
- un projet principal de la fondation Apache
- écrit en langage Java, et peut donc s'exécuter via la machine virtuelle Java sur n'importe quel système d'exploitation
- Aujourd'hui : Tomcat 9

Tomcat n'est pas un serveur Web, et est utilisé en parallèle d'un serveur Web classique (Apache).

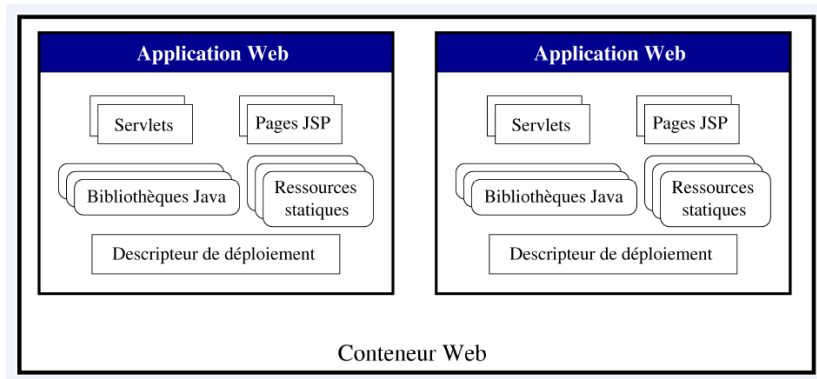
Servlet

Les Servlets : Définition

- Applications java
- Exécutées côté serveur
- Traitement de requêtes HTTP et génération de réponses dynamiques
 - comparables aux CGI
- Permettent d'étendre les fonctionnalités de base d'un serveur HTTP

Avantages

- Portabilité (java)
- Puissance avec l'accès à la totalité de l'API java
- Rapidité (la Servlet est chargée une seule fois)
- Bons outils de développement (Eclipse)
- Compatible avec vos connaissances :)



Servlet : Exemple

Développement de la classe *Hello.java* qui contient la servlet :

HelloWorld.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Hello extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException,
        IOException
    {
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        out.println("Hello, hello !!!!");
    }
}
```

Servlet : Exemple

Routage de la servlet vers l'URL *bonjour*

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>Bonjour</servlet-name>
    <servlet-class>Hello</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Bonjour</servlet-name>
    <url-pattern>/Servlet/Coucou</url-pattern>
  </servlet-mapping>
</web-app>
```

Servlet : Exemple

Exportation de l'application Web dans un fichier d'archive de type *WAR* :

Arborescence

```
ApplicationWeb.war
|_ fichiers.html
|_ fichiers.jsp
|_ répertoires/fichiers
|_ META-INF
    |_ MANIFEST.MF
|_ WEB-INF
    |_ web.xml
    |_ classes
    |_ Servlets.class
    |_ lib
        |_ bibliotheques.jar
```

Servlet : Exemple

Déploiement sur le serveur TOMCAT (via l'interface de manager de TOMCAT)

The Apache Software Foundation
http://www.apache.org/

Gestionnaire d'applications WEB Tomcat

Message: OK

Gestionnaire

[Lister les applications](#) [Aide HTML Gestionnaire](#) [Aide Gestionnaire](#) [Etat du serveur](#)

Applications

Chemin	Version	Nom d'affichage	Fonctionnelle	Sessions	Commandes
/	None specified		true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/j328	None specified		true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/TestWeb	None specified		true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes

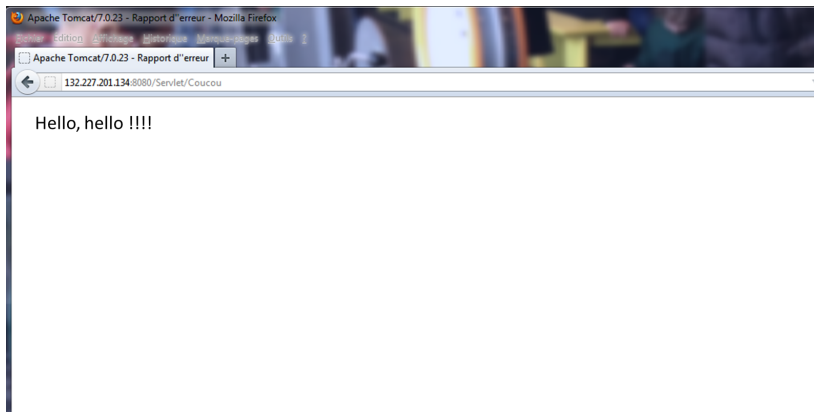
Deployer

Emplacement du répertoire ou fichier WAR de déploiement sur le serveur

Chemin de contexte (requis):

Servlet : Exemple

Appel de la Servlet :



Les servlets peuvent "rendre" n'importe quel type de contenu :

- texte
- HTML
- XML
- binaire (fichier, image, son)
- ...
- JSON

La classe HttpServlet

HttpServlet

```
public abstract class HttpServlet extends GenericServlet
{
    HttpServlet()

    // méthodes répondant aux différents types de requêtes
    protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    protected void doHead(HttpServletRequest req, HttpServletResponse resp)
    protected void doOptions(HttpServletRequest req, HttpServletResponse resp)
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
    protected void doPut(HttpServletRequest req, HttpServletResponse resp)
    protected void doTrace(HttpServletRequest req, HttpServletResponse resp)

    // permet d'utiliser le cache côté client
    protected long getLastModified(HttpServletRequest req)

    // gèrent le dispatching en fonction du type de requête}
    protected void service(HttpServletRequest req, HttpServletResponse resp)
    protected void service(ServletRequest req, ServletResponse res)
}
```

Récupération des paramètres

Les paramètres de formulaires sont récupérables via `HttpServletRequest`. La récupération se fait par les méthodes suivantes :

- *Enumeration* `getParameterNames()`
- *String* `getParameter(Stringname)`
- *String[]* `getParameterValues(Stringname)`

Paramètres de Servlet

```
public class HelloPost extends HttpServlet
{
    public void doGet(HttpServletRequest requete, HttpServletResponse reponse)
        throws ServletException, IOException
    {
        String prenom = requete.getParameter("prenom");
        reponse.setContentType("text/html");
        PrintWriter out= reponse.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Bonjour " + prenom + " ! </h1>");
        out.println("<p>Ceci est ma premiere Servlet...</p>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

URL associée :

<http://.../MyServlet/Hello&prenom=michel>

API REST

API REST (Wikipedia)

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

- Repose sur une architecture client-serveur
- 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.
- Système de couches : chaque composant voit uniquement les composants de la couche avec laquelle il interagit directement
- Utilisation des standards hypermedia : HTML ou XML ou **JSON**

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

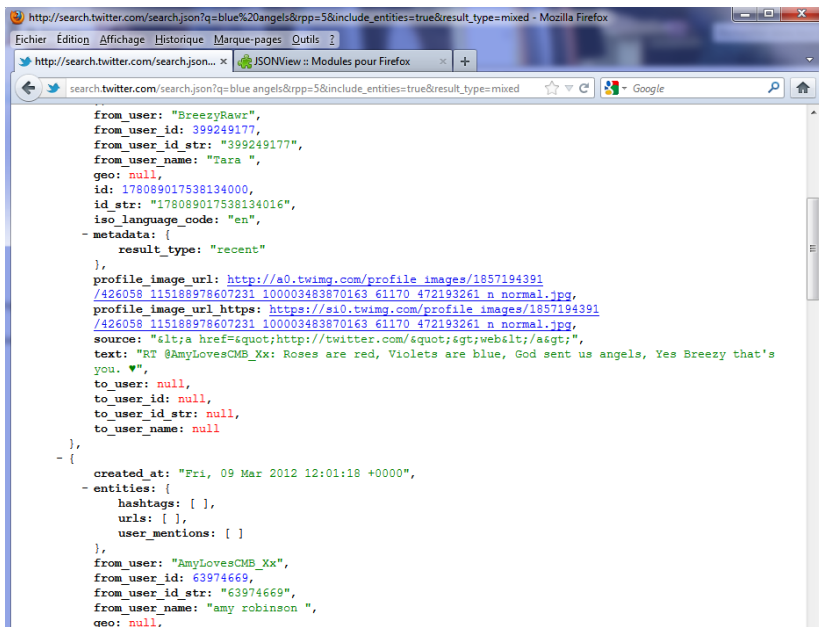
Avantages

- 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)

API REST : Exemple Twitter



```
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://s10.twimg.com/profile\_images/1857194391/426058\_115188978607231\_100003483870163\_61170\_472193261\_n\_normal.jpg,
source: "<a href='\"http://twitter.com/\"'>&gt;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,
```

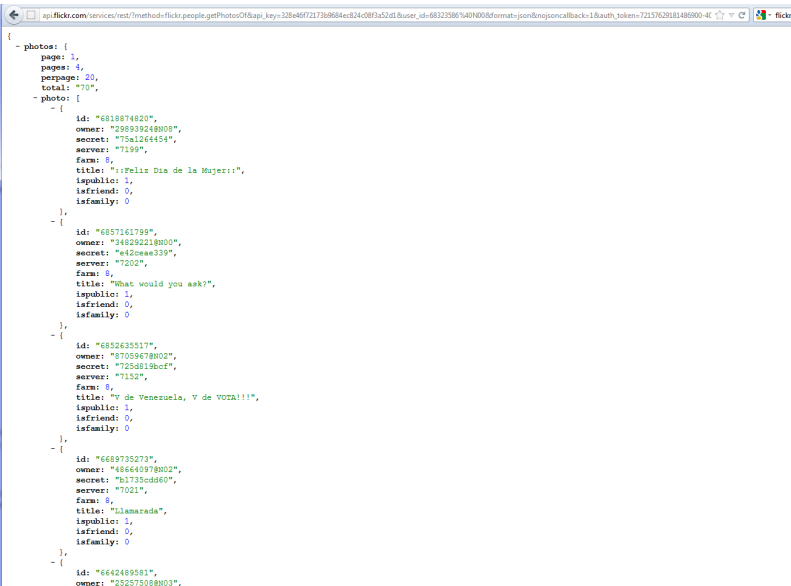

API REST : Exemple Flickr

api.flickr.com/services/rest/?method=flickr.people.getPhotosOf&api_key=328e60f72173b6684ec824c08fa52d1&user_id=68323586%40N00&format=rest&auth_token=72157629181486900-40a224ee68e4346&aj

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="725d819bcf" 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="f867c83f2d" 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="6587896631" owner="8332135@N03" secret="63cc7b496a" server="7142" farm="8" title="The sky is the limit.:" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6586167143" owner="63307805@N00" secret="cfaeb6d377" 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="6566465059" owner="23523125@N08" secret="4c73a0eb85" 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="51/52- 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="20b62858fe" server="7002" farm="8" title=":" ispublic="1" isfriend="0" isfamily="0"/>
    <photo id="6538048271" owner="37763325@N08" secret="66c295ff74" 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="2051f88c38" server="6019" farm="7" title="Friday Favorites" ispublic="1" isfriend="0" isfamily="0"/>
  </photos>
</rsp>
```

API REST : Exemple Flickr



The screenshot shows a web browser window with the address bar displaying a REST client URL for the Flickr API. The main content area shows a JSON response for a GET request to the 'photos' endpoint. The response is a JSON object with a 'photos' array containing metadata and a list of photo objects. Each photo object includes fields like 'id', 'owner', 'secret', 'server', 'farm', 'title', 'ispublic', 'isfriend', and 'isfamily'.

```
{
  - photos: [
    page: 1,
    pages: 4,
    perpage: 20,
    total: "70",
    - photo: [
      - {
        id: "6818874820",
        owner: "296939248N08",
        secret: "75a1264454",
        server: "7199",
        farm: 8,
        title: "¡¡Feliz Día 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: "725d819b0f",
        server: "7152",
        farm: 8,
        title: "V de Venezuela, V de VOTA!!!",
        ispublic: 1,
        isfriend: 0,
        isfamily: 0
      },
      - {
        id: "6689735273",
        owner: "486640978N02",
        secret: "b1735cdd60",
        server: "7021",
        farm: 8,
        title: "Llamarada",
        ispublic: 1,
        isfriend: 0,
        isfamily: 0
      },
      - {
        id: "6642489581",
        owner: "252575088N03",

```

API REST : Exemple Flickr

www.flickr.com/services/api/

- JSON
- PHP

Kits API

Remarque : Les kits API ne sont pas entretenus par Flickr et Flickr n'endosse aucune responsabilité quant à leur utilisation. Pour obtenir de l'aide sur le kit API, rejoignez la [liste de diffusion des développeurs](#) ou contactez directement les personnes chargées de la maintenance.

ActionScript

- [api flickr \(docs\)](#)
- [Flashr](#)
- [Interfaces REST de l'API Flickr](#)
- [as3 flickr lib](#)

C

- [Flickcurl](#)

Cold Fusion

- [CFlickr](#)

Common Lisp

- [Clickr](#)

cUrl

- [Curlr](#)

Delphi

- [dFlickr](#)

Go

- [go-flickr](#)

Java

- [flickrj](#)
- [flickr-jandroid](#)
- [jickr](#)

.NET

- [Flickr.NET](#)

Objective-C

- [ObjectiveFlickr](#)

Perl

- [Flickr-API2](#)
- [Flickr:Upload](#)

PHP

- [PEAR: Flickr_API](#)
- [phpFlickr](#)

PHP5

- [Phlickr](#)

Python

favorites

- [flickr.favorites.add](#)
- [flickr.favorites.getContext](#)
- [flickr.favorites.getList](#)
- [flickr.favorites.getPublicList](#)
- [flickr.favorites.remove](#)

galleries

- [flickr.galleries.addPhoto](#)
- [flickr.galleries.create](#)
- [flickr.galleries.editMeta](#)
- [flickr.galleries.editPhoto](#)
- [flickr.galleries.editPhotos](#)
- [flickr.galleries.getInfo](#)
- [flickr.galleries.getList](#)
- [flickr.galleries.getListForPhoto](#)
- [flickr.galleries.getPhotos](#)

groups

- [flickr.groups.browse](#)
- [flickr.groups.getInfo](#)
- [flickr.groups.search](#)

groups.members

- [flickr.groups.members.getList](#)

groups.pools

- [flickr.groups.pools.add](#)
- [flickr.groups.pools.getContext](#)
- [flickr.groups.pools.getGroups](#)
- [flickr.groups.pools.getPhotos](#)
- [flickr.groups.pools.remove](#)

interestingness

- [flickr.interestingness.getList](#)

machinetags

- [flickr.machinetags.getNamespaces](#)
- [flickr.machinetags.getPairs](#)
- [flickr.machinetags.getPredicates](#)
- [flickr.machinetags.getRecentValues](#)
- [flickr.machinetags.getValues](#)

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.

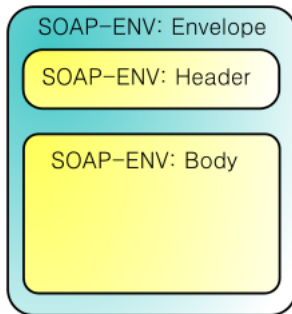
105: Service currently unavailable

The requested service is temporarily unavailable.

111: Format "xxx" not found

The requested response format was not found.

REST vs SOAP



©<https://fr.wikipedia.org/wiki/SOAP>

Le protocole SOAP est composé de deux parties :

- une enveloppe, contenant des informations sur le message lui-même afin de permettre son acheminement et son traitement ;
- un modèle de données, définissant le format du message, c'est-à-dire les informations à transmettre.

REST vs SOAP

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

JSON

JSON

- 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

JSON

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.

JSON

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

JSON

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

JSON

```
{
  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 }
}
```

JSON et JAVA

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, REST et JSON

```
public class Example extends HttpServlet
{
    .....

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException
    {
        response.setContentType("text/plain");
        String prenom=request.getParameter("prenom");
        if (prenom==null)
        {
            JSONObject json=new JSONObject();
            try {
                json.put("error","Missing parameter");
            } catch (JSONException e) {
                e.printStackTrace();
            }
            response.getWriter().println(json.toString());
        }
        else
        {
            JSONObject json=new JSONObject();
            try {
                json.put("output","OK");
            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            response.getWriter().println(json.toString());
        }
    }
}
```