

3IN017 - TECHNOLOGIES DU WEB

Introduction - Architecture orientée service

17 janvier 2023

Gilles Chagnon



Organisation de l'UE

Acquisition de techniques pour le développement de sites Web « modernes »

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

Objectifs de l'UE

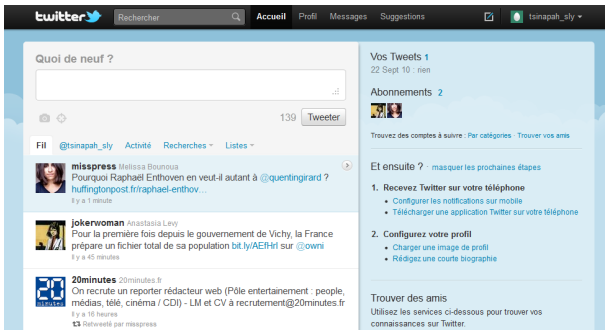
Un enseignement « concret » basé sur la manipulation de technologies

- Cours : Présentation des différentes technologies et de leur articulation, cours d'ouverture sur des sujets connexes
- TD : Prise en main des technologies
- TP : Mise en œuvre de ces technologies

Les TPs sont tous structurés autour du développement d'un site web

Objectifs de l'UE

Développement d'un site web type social



Évaluation 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
 - Évaluation serveur à mi-parcours, document à rendre après le TME5
 - Soutenance de projet par vidéo
 - TME solo
- 50% : Examen terminal sur feuille. L'examen est long. . .

Évaluation serveur à mi-parcours

- Document à rendre après le TM5
- Tableau de services modifié (consignes à venir)

Soutenance de projet

- Projet en binôme
- Réalisation de Birdy côté Client + côté Serveur
- Implémentation de fonctionnalités **obligatoires** - voir cahier des charges
- Implémentation de fonctionnalités **additionnelles**

Notation :

- Présentation de la réalisation (en vidéo)
- Modifications à apporter au projet (TME solo)

Attention : il y a du code existant, on le sait et on va vérifier :

- Vérification automatique de plagiat (avec projets existants sur le web et tous les projets de la promo)
- Vous devez connaître parfaitement votre code
- Vous devez savoir l'expliquer, et y ajouter des modifications

À réaliser :

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

Planning de l'UE

S	Cours	TME1	TME2
1	Intro - AOS/Services	-	-
2	HTML/CSS	Modélisation	Prise en main / API REST
3	Javascript		HTML/CSS
4	React JS		Javascript
5	React JS		ReactJS
6	NodeJS		ReactJS
7	NodeJS		NodeJS
8	NoSQL		NodeJS
9	Accessibilité numérique		NoSQL
10	Mise en production, cloud		fin serveur
11	cours d'ouverture		fin serveur
12	-		TME solo

Attention :

On n'a que 11 semaines ⇒ Il faut utiliser des technos récentes et robustes.

Concrètement

Ce que l'on va utiliser :

- Présentation graphique
 - HTML, CSS, Javascript/React JS
- Serveur Web
 - Node
- Language développement côté serveur
 - Nodejs (javascript)
- Communication client - serveur
 - React JS (librairie axios)
- Bases de Données
 - SQLite (similaire à MySQL/Oracle/...), NedB (basé sur MongoDB)

Concrètement

Étude 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

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

Cela implique :

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

Objectif double du cours

- Vous enseigner – donner un point d'entrée – pour un ensemble de technologies
- 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)

En résumé...

Objectif double du cours

- Vous enseigner – donner un point d'entrée – pour un ensemble de technologies
- Vous « éclairer » sur les évolutions actuelles de l'informatique

Ce que le cours n'est pas/ne vous apporte pas

- Une connaissance profonde de toutes les technologies
- Vous ne serez *pas* expert de développement web à la fin de ce cours. A vous de parfaire vos connaissances
- Il y a des technologies différentes (Spring, Angular, Symfony...) qui demandent un niveau d'abstraction plus important. Si vous comprenez ce cours, vous saurez apprendre ces technologies.

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 pouvez (devez) 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=qcot5k95xffzxpdl1dznekc bhao

Questions Cours/TD

Logistique - Examens

Random - ActuWeb

Grp1

Grp2

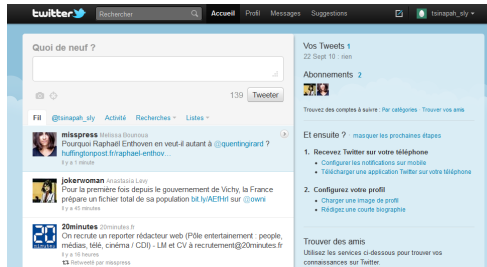
...

- Gilles Chagnon : responsable d'UE, chargé de cours, TD/TME
- Benjamin Becquet : chargé de cours, TD/TME
- Sébastien Lataix : chargé TD/TME
- Alexy Morcillo : chargé TD/TME
- Tanguy Soto : chargé TD/TME

Questions ?

Développement web

Site web ?



Développer un site web ? Facile !

- Systèmes de gestion de contenu (Content Manager System CMS) : généralistes (Wordpress, Joomla, Drupal, Google sites, Wix, Mediawiki. . .) ou spécialisés (Prestashop, LMS comme Moodle, groupwares comme Nextcloud ou Microsoft 365. . .)
- Hébergeurs de site web : en France OVH, Hostinger, LWS. . .

Mais. . .

Développer un site web c'est aussi maîtriser son développement, sa maintenance et utiliser les technologies adaptées en fonction des besoins de ses utilisateurs !

Un peu d'histoire... La première page web

- CERN – 1989 : "World Wide Web" (URL, HTML, HTTP)

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 r](#)

[What's out there?](#)

Pointers to the world's online information, [subjects](#) , [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#) ,X11 [Viola](#) , [NeXTStep](#) , [Servers](#) , [Tools](#) , [Mail robot](#) , [Library](#))

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help ?](#)

If you would like to support the web..

[Getting code](#)

Getting the code by [anonymous FTP](#) , etc.

`http://info.cern.ch/hypertext/WWW/TheProject.html`

- premier navigateur web : `https://worldwideweb.cern.ch/`


Un peu d'histoire...

- fin des années 60 : ARPANET
- 1971 : 23 ordinateurs reliés
- 1973 : définition du protocole TCP/IP
- 1983 : adoption de TCP/IP et du mot « Internet »
- 1990 : World Wide Web, HTTP, 100 000 ordinateurs
- 1992 : 1 000 000 d'ordinateurs
- 1993 : HTML
- 1994-1995 : Netscape, Yahoo!, Amazon, PHP, eBay, Internet Explorer, JavaScript, MySQL
- 1996 : CSS
- 1998 : Google
- 2000 : Éclatement de la Bulle Internet (368 000 000 ordinateurs)
- 2001 : Wikipédia
- 2002-2006 : Firefox, Wordpress, Facebook, YouTube, Twitter...
- 2008 : Google Chrome
- 2009 : MongoDB
- 2010 : Instagram, Pinterest
- 2014 : 1 milliard de sites Web
- 2016 : Tiktok
- 2021 : 4,6 milliards d'ordinateurs

Un peu d'histoire... 1994 : W3C

W3C

The World Wide Web Consortium (W3C) is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards. Led by Web inventor and Director Tim Berners-Lee and CEO Jeffrey Jaffe, W3C's mission is to lead the Web to its full potential. (<https://www.w3.org/Consortium/>)


Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents

Validate by URI Validate by File Upload **Validate by Direct Input**

Validate by direct input
Enter the Markup to validate:

▼ More Options

- ☒ **Validate Full Document**
Use Doctype: (detect automatically) ☐ Only if Doctype is missing
- ☐ **Validate HTML fragment**
Use Doctype: ☒ HTML 4.01 ☐ XHTML 1.0
- ☒ List Messages Sequentially ☐ Group Error Messages by Type
- ☐ Show Source ☐ Clean up Markup with HTML-Tidy
- ☐ Show Outline ☐ Validate error pages ☐ Verbose Output

Check

Site statique vs. dynamique

- Site statique : contenu de la page est figé (HTML, CSS)
 - Site dynamique : contenu non figé
 - Adaptation à l'utilisateur, ses préférences
 - Scripts qui permettent de changer de contenu (PHP, JavaScript, Java, Python, Perl...)
 - Affichage varie (CSS, Javascript)
 - Syndication et agrégation de contenus (API...) : météo, actus, publicité...
- Vers une réutilisation des composants : on crée des *applications* web !

Pourquoi est-ce compliqué de développer des sites web ?

- De nombreuses technologies :
 - Document : DOM, HTML...
 - Mise en forme : CSS
 - Interaction côté client : Javascript
 - Interaction côté serveur : Java, PHP, Python, nodeJS...
 - Bases de données : MySQL, SQLite, Oracle... NoSQL, MongoDB, HBase...
- des technologies en constante évolution
- d'autres technologies : frameworks (Spring, Symfony, Laravel, Next.js... côté serveur ; (jQuery), Angular, AngularJS, VueJS, React, Svelte... côté client), CMS, sécurité, réseau...

→ **Veille technologique constante**

Quelles données sur le web ?

- Diffusion et partage de connaissances : broadcasting, wikipédia, vidéos, podcasting. . .
- réseaux sociaux, forums
- Site web d'entreprise, pages personnelles : question de perception !
Attention à ce que vous mettez sur le web ! (recrutement : données, clarté, esthétique. . .)

Notions de base

Hypertext Transfer Protocol (HTTP)

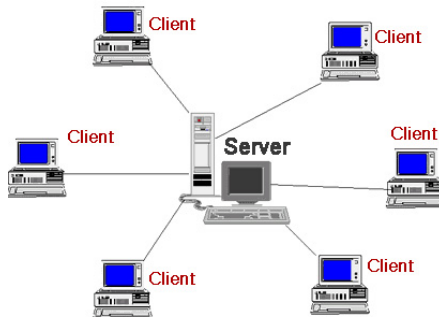
- Protocole inventé en 1989 au CERN par Sir Tim Berners-Lee
- Il détermine comment les informations sont transmises entre le client et le serveur
- Plusieurs méthodes de transmission (requêtes) :
 - GET – la plus courante – demander une ressource.
 - POST – courante également – transmettre des données en vue d'un traitement à une ressource (création de nouvelles ressources et modification).
 - PUT – courante également – remplacer ou ajouter une ressource sur le serveur.
 - PATCH – modification partielle d'une ressource.
 - DELETE – supprimer une ressource du serveur
 - OPTIONS – options de communication d'une ressource ou du serveur en général.
 - CONNECT – utiliser un proxy comme un tunnel de communication.
 - TRACE – demande au serveur de retourner ce qu'il a reçu, dans le but de tester et effectuer un diagnostic sur la connexion.

URL

- La transmission des informations est effectuée via l'URL (Uniform Resource Locator)
- Format standardisé :

Protocole	[Mot de passe]	Nom du serveur	[Port] (facultatif si 80)	Chem
http ://	user :password@	www.monserveur.fr	:80	/logi

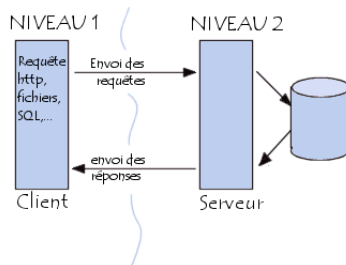
Architecture Client-Serveur



Mais plusieurs configurations :

- Un client / un serveur
- Plusieurs clients / un serveur
- Un client / des serveurs

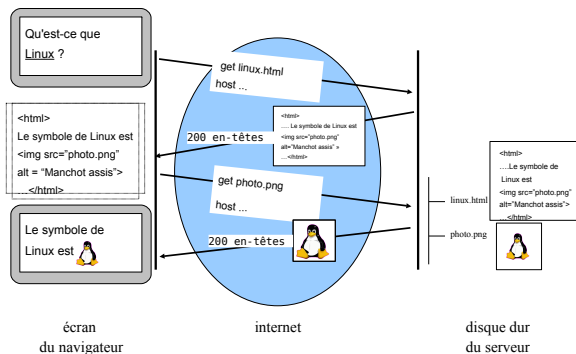
Architecture Client-Serveur



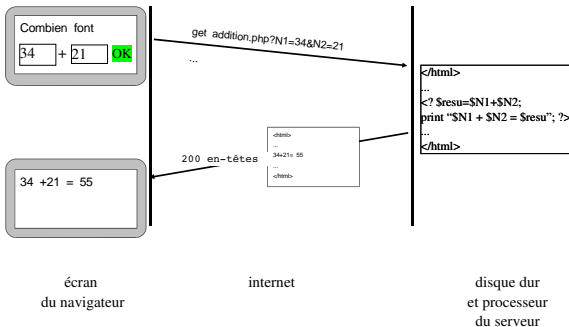
- Client (HTML, CSS, JavaScript...)
 - Il établit la connexion au serveur à destination d'un ou plusieurs ports réseau
 - lorsque la connexion est acceptée par le serveur, il communique /interroge le serveur au moyen de **requêtes**.
- Serveur
 - Il attend une connexion entrante sur un ou plusieurs ports
 - À la connexion d'un client sur le port, il communique avec le client au moyen de **réponses**.

Architecture Client-Serveur : chargement pages web

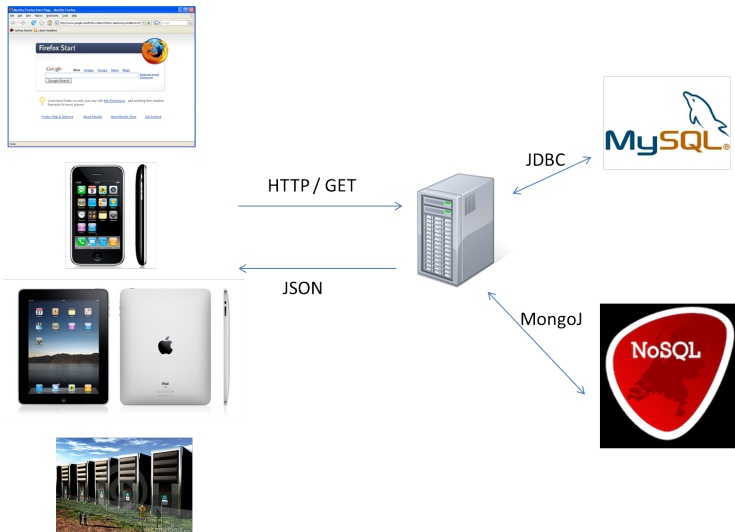
Pages Web stockées sur un serveur Web



Mise en place de services Web



Web dynamique / Web API



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 wikipédia)

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 wikipédia)

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{...}
```


Différents paradigmes :

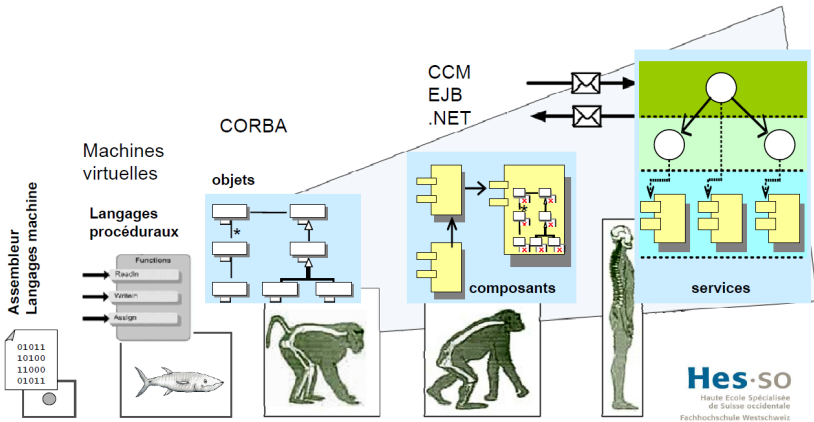
- Procédures
- Programmation Orientée Objet
- Programmation Orientée Composants

(source wikipédia)

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

- 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*

Paradigmes de programmation : Programmation Orientée Composants

Différents paradigmes :

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

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.

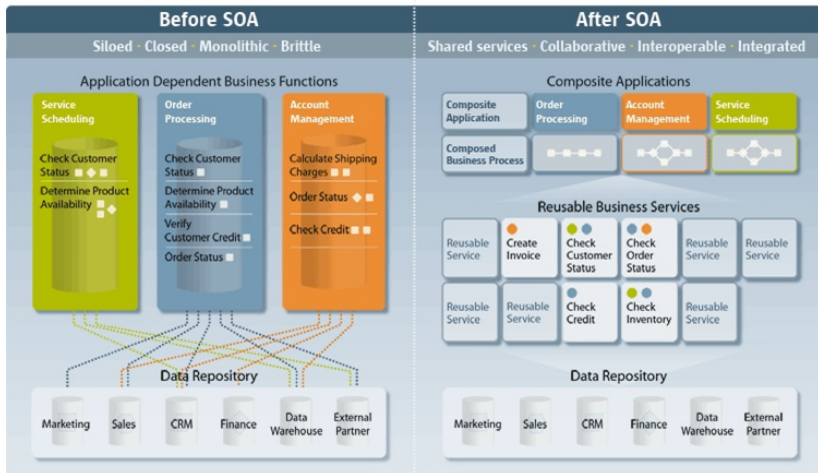
Lorsque l'architecture SOA s'appuie sur des web services, on parle alors de WSOA, pour **Web Services Oriented Architecture**.

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

- Coûts de conception et de développement initiaux conséquents
- Les services Web souffrent de performances faibles pour des traitements faibles (notion de couches)



Les Web Services

- Un Service est Autonome (et sans état)



- Les Frontières entre services sont Explicites



Source A. Ocella

- Un Service expose un Contrat

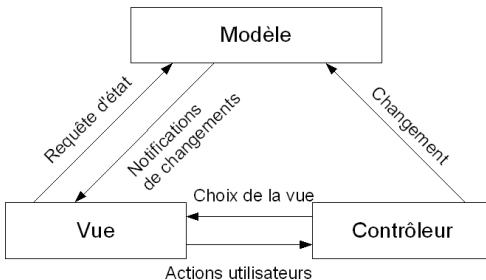


Conditions Générales de Vente
Règlement Intérieur
Vos droits/Vos devoirs

- Les services communiquent par messages



Modèle - Vue - Contrôleur (MVC)



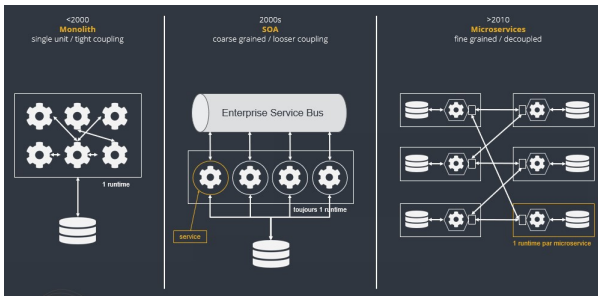
Découpage logiciel : MVC vs. SOA

- Histoire de timing : SOA, puis MVC (puis microservices)
- MVC : pas une pleine autonomie entre les fonctionnalités ("services" en SOA), découpage guidé.
- SOA : autonome, seul le partage des schémas ("modèle" en MVC), services plus portables.
- SOA : découpage au niveau services entre développeurs

Microservices

Microservices : découpage logiciel ET infrastructures

- Un découpage plus petit des « services »
- Un découpage au niveau infrastructure : chaque microservice a son infrastructure
 - Optimisation par microservice
 - Plus de flexibilité dans l'évolution, évolution accélérée
 - Résolution plus rapide des problèmes



©<https://softeamgroup.github.io/microservices-state-of-play/>

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
- Évolutivité
- Repose sur les principes du Web
- Représentations multiples

Inconvénients

- Sécurité restreinte par l'emploi de HTTP (il faut ajouter le chiffrement TLS)
- Cible uniquement l'appel de ressources
 - Architecture orientée ressources (ROA)
 - ou Architecture orientée données (DOA)

Exemple d'accès aux ressources avec API REST (« naming »)

- Une ressource : équivalent de l'objet en java (raccourci très rapide...). Une "chose" et non une "action". Par ex : produit, utilisateur, ...
- Une ressource peut être un **singleton** (customer) ou une **collection** (customers).
- Accès à une ressource (appelé **document**) de la collection :
/customers/customerId,
`http://api.example.com/user-management/users/{id}`.
- Une ressource peut contenir une collection de ressources :
/customers/customerId/accounts. On appelle cela un **store**.
- on peut appeler des actions (appelées **controllers**) sur la ressource : `http://api.example.com/song-management/users/{id}/playlist/play`

Pour plus d'information : <https://restfulapi.net/resource-naming/>

Exemple d'accès aux ressources avec API REST (« naming »)

- On utilise ensuite le protocole HTTP pour appeler les actions CRUD :

HTTP GET

`http://api.example.com/device-management/managed-devices`
//Get all devices

HTTP POST

`http://api.example.com/device-management/managed-devices`
//Create new Device

HTTP GET

`http://api.example.com/device-management/managed-devices/id`
//Get device for given Id

HTTP PUT

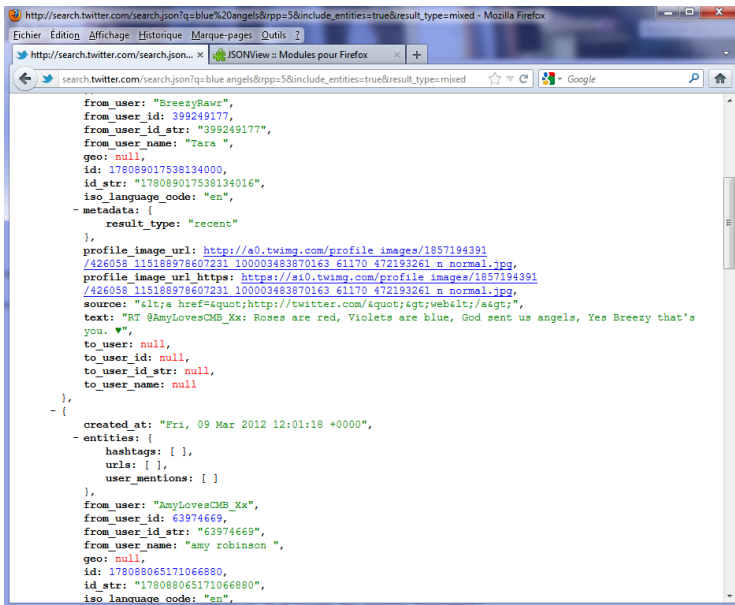
`http://api.example.com/device-management/managed-devices/id`
//Update device for given Id

HTTP DELETE

`http://api.example.com/device-management/managed-devices/id`
//Delete device for given Id

Pour plus d'information : <https://restfulapi.net/resource-naming/>

API REST : Exemple Twitter v1 (maintenant fermée)



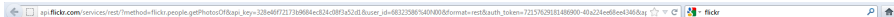
```

http://search.twitter.com/search.json?q=blue%20angels&rpp=5&include_entities=true&result_type=mixed - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils ?
http://search.twitter.com/search.json... x JSONView :: Modules pour Firefox
search.twitter.com/search.json?q=blue angels&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://s10.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



```


- <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="e42ceae339" server="7202" farm="8" title="What would you ask?" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="6852635517" owner="8705967@N02" secret="725d819bcb" 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="961ed24366" server="7165" farm="8" title="2011 in Review" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="6593866295" owner="38838924@N03" secret="B67c8324d" 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="6586167143" owner="63307805@N00" secret="cfab6d377" 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="473a0a6b85" server="7147" farm="8" title="Feliz Navidad..." ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="6564959719" owner="29219049@N06" secret="420005b9d4" server="7153" farm="8" title="Feliz Navidad..." ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="6560620287" owner="40654531@N05" secret="80da0970a" server="7001" farm="8" title="51.52: Feliz Navidad!!!" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="6553379789" owner="49762063@N08" secret="20b628586" server="7002" farm="8" title="mi navidad, tu navidad? Explored Dec 23, 2011 #6 Muchas Gracias!" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="6538048271" owner="3776323@N08" secret="66c295874" server="7171" farm="8" title="Felices Fiestas! ~ Happy Holidays!" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="6526003925" owner="66881369@N08" secret="ebb8093294" server="7020" farm="8" title="Family lunch" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="6475299671" owner="43616712@N03" secret="aa811060f7" 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

```
api.flickr.com/services/rest/?method=flickr.people.getPhotosOf&api_key=328e86f72173b9684ec824c08f3a52d&user_id=68323586%40N00&format=json&nojsoncallback=1&auth_token=72157629181486900-4c
{
  "photos": {
    "page": 1,
    "pages": 4,
    "perpage": 20,
    "total": "70",
    "photo": [
      - {
        id: "6818874820",
        owner: "298593240N00",
        secret: "75a1264454",
        server: "7199",
        farm: 8,
        title: "¡Feliz Día de la Mujer!",
        ispublic: 1,
        isfriend: 0,
        isfamily: 0
      },
      - {
        id: "6857161799",
        owner: "348292210N00",
        secret: "642ceae339",
        server: "7202",
        farm: 8,
        title: "What would you ask?",
        ispublic: 1,
        isfriend: 0,
        isfamily: 0
      },
      - {
        id: "6852635517",
        owner: "87059670N02",
        secret: "725d819bcf",
        server: "7152",
        farm: 8,
        title: "V de Venezuela, V de VOTA!!!",
        ispublic: 1,
        isfriend: 0,
        isfamily: 0
      },
      - {
        id: "6609735273",
        owner: "486640970N02",
        secret: "2b1735cdd60",
        server: "7021",
        farm: 8,
        title: "Llamarada",
        ispublic: 1,
        isfriend: 0,
        isfamily: 0
      },
      - {
        id: "6642489581",
        owner: "252575080N03",
        secret: "a4609aec6d",
        server: "7025",
        farm: 8,
        title: "Things I Love Thursday",
        ispublic: 1
      }
    ]
  }
}
```

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

- Flickrurl

Cold Fusion

- CFlickr

Common Lisp

- Clickr

cURL

- Curlr

Delphi

- dFlickr

Go

- go-flickr

Java

- flickrj
- flickr-jandroid
- jlickr

.NET

- Flickr.NET

Objective-C

- ObjectiveFlickr

Perl

- Flickr-API2
- Flickr::Upload

PHP

- PEAR::Flickr_API
- phpFlickr

PHP5

- Phlickr

Python

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

The SOAP envelope send in the request could not be parsed.

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.
- Formellement proche du XML, mais moins verbeux

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 }
}
```

Next step ?

Quelles technologies ?

Ce que l'on va utiliser :

- Présentation graphique
 - HTML, CSS, Javascript/React JS
- Serveur de service Web
 - Node
- Language développement côté serveur
 - Nodejs (javascript)
- Communication client - serveur
 - React JS (librairie axios)
- Bases de Données
 - SQLite (similaire à MySQL/Oracle/...), NedB (basé sur MongoDB)

Semaine prochaine. . .

JAVASCRIPT PARTOUT !!! C'est la base
Prochain cours : HTML/CSS !

- TD 1 : Modélisation d'un site web
- TME1 : Installation de l'environnement, premiers codes. . .