Technologies du web

7 - Web services & AJAX

Architectures web

D’abord de simples pages HTML, le web s’est complexifié pour proposer aujourd’hui des interfaces très évoluées : les **applications web**.

Elle remplace aujourd’hui les applications de bureaux

Architecture web en 90

Serveur de pages HTML statiques.

Pas de Javascript

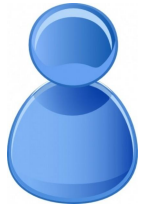
Les pages sont créées/éditées à la main ou avec des logiciels dédiés (exemple Dreamweaver)

Web 1.0 ou web passif

Le contenu servi est issu de données stockées sur le serveur.

Copie de fichiers par FTP

concepteur



Architecture web en 2000

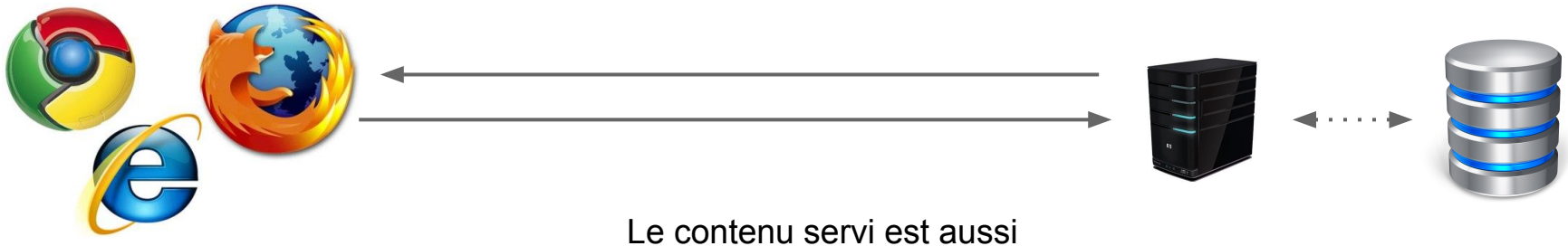
Apparition du web 2.0

Le contenu est généré par les utilisateurs via des **formulaires**

Les pages sont générées **dynamiquement** en fonction du contenu posté (**PHP**, etc…)

Utilisation de CMS / Wiki / etc ...

Web 2.0



Le contenu servi est aussi fourni en partie par les utilisateurs via le site web

Architecture web 2010

Les clients et les serveurs web s'échangent des données via des **webservices**.

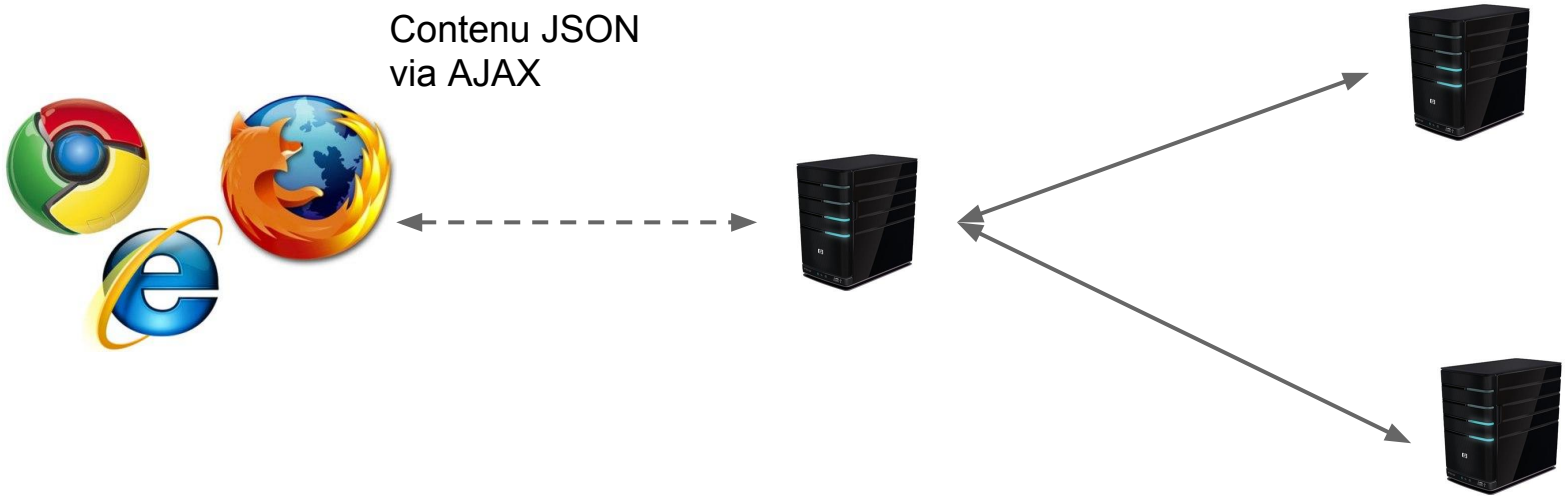
Échange de données entre serveurs web

Utilisation intensive de Javascript pour rendre les pages **dynamiques côté client** (AJAX)

Application web

Contenu HTML
Contenu JSON
via AJAX

Appels de webservices
entre serveurs



Web service

Programme permettant la communication et l'**échange de données entre machines distantes** et fonctionnant sur les protocoles du web (à savoir HTTP) et ce, généralement de manière **asynchrone**.

Les données envoyées par un web service sont destinées à **être utilisées par un programme informatique** et non pas par un humain comme avec le HTML.

Fournir des données "Machine Readable" ou Communication "machine à machine"

Architecture SOA

Service Oriented Architecture

L’application est décomposée en différents programmes qui sont exécutés sur différentes machines.

Les machines doivent donc communiquer entre elles pour :

- échanger des données
- appeler des procédures distantes

Implementation SOA

Initialement proposé par des technos comme CORBA / RMI / DCOM

=> couplage fort entre les applications.

Les services web utilisent HTTP pour le transport et des formats de données comme XML ou JSON

=> **couplage faible**.

Caractéristiques d’un service web

- URL d’appel
  - identification de la procédure à appeler
- un format de données
  - généralement JSON ou XML

Pas de langage de programmation spécifique.

Fournisseur de web services

Service de recherche, d'accès ou en encore de transformation de données

Annuaire de personnes, authentication, site open-data, geocoding, agence de voyages, etc…

Les fournisseurs de WS proposent **une API (application programming interface)** qui décrit les services disponibles

Utilisation des web services

- Par un serveur web qui a besoin d'informations externes à ajouter dans son site
- Par les applications (mobiles entre autre) qui ont besoin de données ou de résultats de calculs distants
- Par une application web fonctionnant dans le navigateur qui veut rafraîchir ses données **sans avoir à regénérer la page entièrement**
- ...

Standards et architectures des WS

SOAP : protocole d’échange de message XML

XML-RPC : protocole XML d’appel de procédure distante

WSDL : Standard de description de service

REST : Architecture de web service sans état utilisant le protocole HTTP.

API

Application programming interface

Définition des **méthodes** ou procédures distantes disponibles sur un serveur.

- Nom de la méthode
- Paramètres
- Résultat
- Format de données

REST

Architecture de web service utilisant HTTP et defini par

- une **URI** pour spécifier la ressource / méthode à appeler et ce **de manière unique**
- Une **action HTTP** (GET, POST, PUT, DELETE…)
- **Absence d'état** du serveur : la même requête donnera toujours le même résultat.

La construction des URI doit permettre de décrire de manière unique la ressource à accéder:

GET http://mon-serveur.org/article/1234553

Exemple appel WS REST

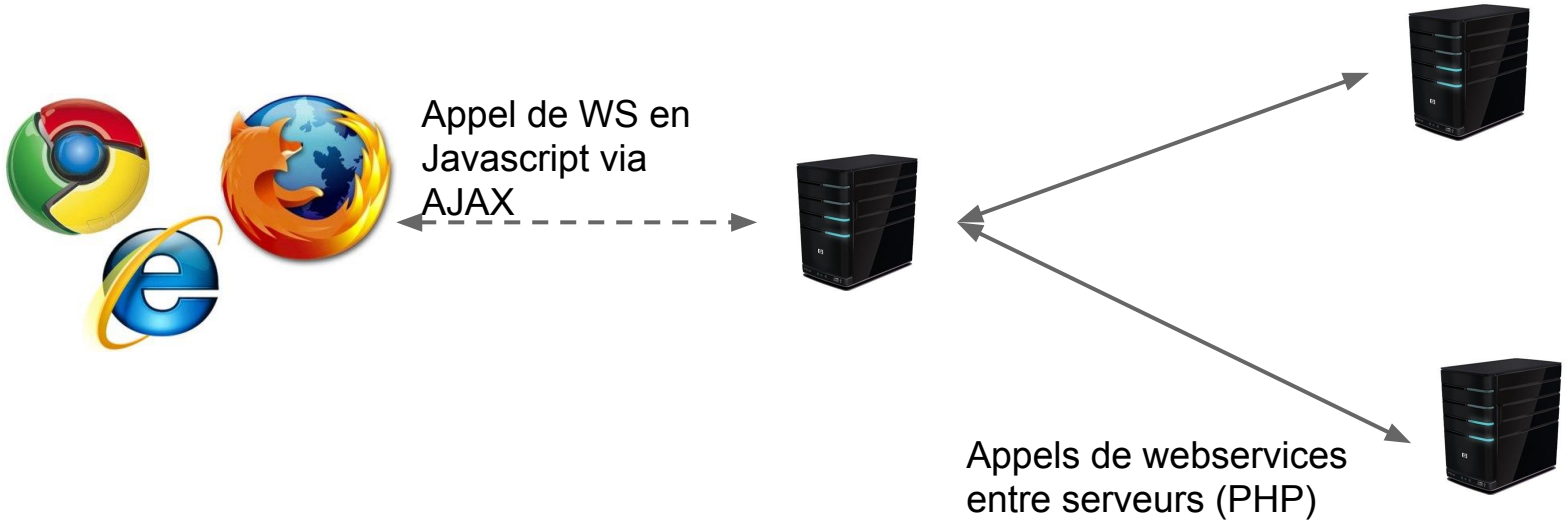
Recherche de livre sur google books

https://www.googleapis.com/books/v1/volumes?q=albert%20camus

Appel de service REST

Consommation / Utilisation d'un web service

- depuis un programme PHP
- depuis le navigateur grâce à Javascript (AJAX)


Appel de WS en Javascript via AJAX
Appels de webservices entre serveurs (PHP)

En PHP

```
// Utilisation de l'extension curl.
$url = 'http://unserveur.fr/api/ressource/12?format=json'
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL, $url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
// Execution de la requete
$result = curl_exec($curl);

curl_close($curl);


$result_array = json_decode($result);
```

En Javascript depuis un navigateur

- Les navigateurs définissent un ensemble de fonctions javascript permettant d'appeler un webservice pour obtenir du XML ou du JSON

- Elles sont regroupées sous l'objet **XMLHttpRequest**

- Cette fonctionnalité à donné naissance à **AJAX**

AJAX

Asynchronous JavaScript and XML

Technologie apparue entre 2002 et 2005

Permet de rendre le contenu d’une page dynamique grâce à l’appel de WS de manière **asynchrone**.

Appel Asynchrone

Le programme n’est pas bloqué et n’attend pas la réponse de la requête immédiatement.

À la place, on définit une fonction spécifique qui sera exécutée lorsque la réponse sera reçue: le **callback**

Comportement sans AJAX

Une requête est envoyée par le navigateur:

- Quand une nouvelle URL est tapée
- Quand on clique sur un lien <a>
- Quand on “submit” un formulaire
- Quand on charge une image <img>
- Quand on charge un fichier javascript <script>

Une **nouvelle page** est retournée par le serveur puis affichée par le navigateur

Comportement avec Ajax

Permet de charger des données distantes, **sans recharger toute la page**. L'appel AJAX a lieu lors d'un événement JS

- click (bouton, lien)
- timer
- etc…

Le code javascript émet une requête asynchrone.

A la réponse, une fonction JS est exécutée : elle analyse alors les données reçues (en JSON) et **modifie** la page courante.

Pourquoi faire ?

Eviter la transmission et l'affichage d'une nouvelle page complète à chaque interaction.

Améliorer la **réactivité** de l’application.

Mettre à jour “en temps réel” la page sans action de l’utilisateur dans son navigateur.

ajax : XMLHttpRequest

En Javascript on lance une requête AJAX avec l'objet **XMLHttpRequest**

```
var request = new XMLHttpRequest();

request.addEventListener('load', function(data) {
      console.log(data);
});

request.addEventListener('error', function(data) {
      console.log('error', data);
});


request.open("GET", "php/mon_ws.php");
request.send();
```

Exemple

Dans une page HTML, on veut faire un flux d’actualité qui se rafraîchit tout seul toutes les 30 secondes.

On utilise un webservice php qui renvoie en JSON ce flux.

Le Web Service (sur le serveur)

http://localhost/api/get_lastest_news.php

Le serveur renvoie du JSON (et non pas du HTML)

```
{ news : ["News 1...", "News 2"] }
```

WS JSON en PHP

```
<?php
// get data
$data = Array("news" => Array ("news 1", "news 2", ...));

// output json
echo json_encode($data);
?>
```

Création d’un web service en PHP

On peut créer un fichier PHP pour chaque service.

GET http://monserveur.org/api/service1.php

GET http://monserveur.org/api/service2.php

Côté client

On va utiliser Javascript pour interroger le web service toutes les N secondes et mettre à jour la page, sans la recharger

HTML

```
<body>
    …
    <aside id="news">
        <ul>
        </ul>
    </aside>
    …
</body>
```

Javascript

```
document.addEventListener('DOMContentLoaded', function () { // après chargement
      function refresh_news() {
         var request = new XMLHttpRequest();

        request.addEventListener('load', function(data) {
            var ret = JSON.parse(data.target.responseText);
            var new_html = '';
            for (var i = 0; i < ret.news.length; i++) {
                 new_html += build_msg_html(ret.news[i]);
            }

            document.querySelector('#messages').innerHTML = new_html;
        });

        request.open("GET", "php/get_latest_msg.php");
        request.send();
       }
      // rafraichissement toutes les 30 secondes
      setInterval(refresh_news, 30000);
});
```

Envoi de formulaire en AJAX

De la même manière, on peut envoyer des données de formulaire de manière asynchrone en faisant une requête POST

Les données sont envoyées en arrière plan sans que la page soit rechargée.

Javascript

```
var form = document.getElementById('msg-form');
form.addEventListener("submit", function(event) {
        event.preventDefault(); // ne pas recharger la page

        var request = new XMLHttpRequest();

        request.addEventListener('load', function(data) {
            console.log(data);
            var textarea = document.getElementById('msg');
            textarea.value = '';
        });

        request.addEventListener('error', function(data) {
            console.log('error', data);
        });

        request.open("POST", "php/add_msg.php");
        request.send(new FormData(form));

    });
```

Restriction d’appel de WS en JS

Pour des raisons de sécurité, les navigateurs interdisent d’appeler des WS dont le nom de domaine est différent de l’origine du code : **“same-origin policy”**

Une application web servie depuis
http://mon-serveur.fr/index.html
ne pourra appeler que des WS sur
http://mon-serveur.fr/...

Ce qu'il faut retenir

Un WebService fournit des données qui seront traitées par un programme.

Un WS REST s'appelle via une URL sur le protocole HTTP.

On peut **utiliser** un WS,

- depuis du code PHP sur le serveur,
- depuis du code Javascript sur le client grâce à AJAX

Ajax permet de rendre les pages HTML plus reactives et dynamiques.

Les WS peuvent **être écrits** en PHP sur le serveur

Conclusion

Les webservices sont largement utilisés aujourd'hui dans les applications web

=> Applications web **plus interactives** et dynamiques

=> Facilite la création d'**architectures distribuées**

=> **Séparation claire** entre l'interface utilisateur et les procédures de gestion de données et de calcul (le métier).