AJAX – VERSION COURTE – V2023

SOMMAIRE

Edition : 2023

AJAX

https://openclassrooms.com/courses/creez-des-pages-web-interactives-avec-javascript/manipulez-les-formulaires

Installation des fichiers de tests

Dans le cours :

Les exemples sont présentés dans un chapitre en vert.

Les exercices à faire sont présentés dans un chapitre en jaune.

1 - Présentation AJAX

http://www.w3schools.com/ajax/default.asp

Historique

Terme inventé en 2005. Le début date d'Internet Explorer 5, en 1999, avec l'introduction de l'**objet XMLHttpRequest** par Internet Explorer.

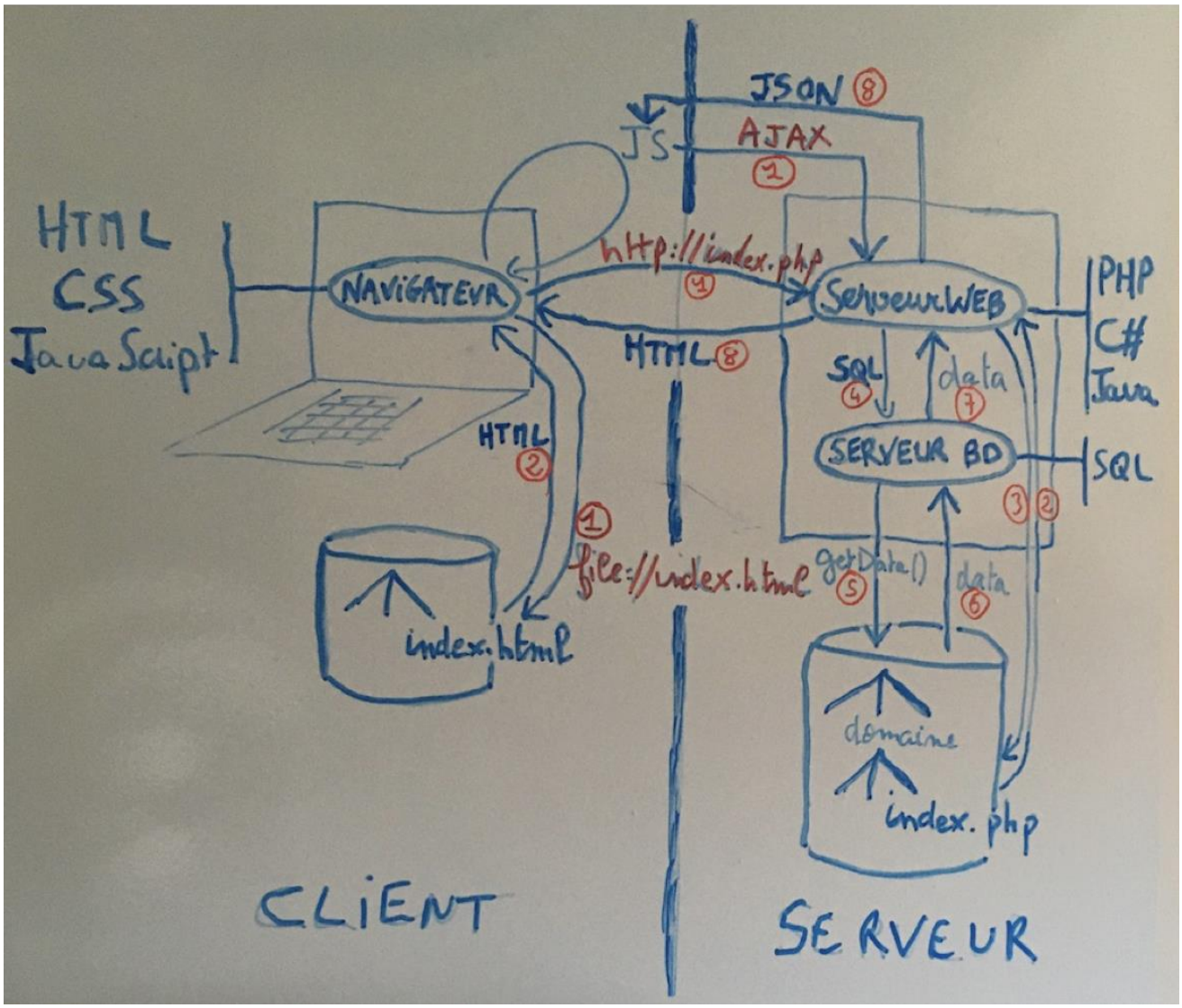
Objectifs

- Ne plus avoir à faire des mises à jour complète d'une page web, mais seulement une partie.
- Avoir des **échanges asynchrones** : le client fait une demande mais n'attends pas la réponse.

AJAX

- Il s'agit d'un ensemble de méthodes intégrées dans JavaScript pour transférer, « en coulisse », des données entre le navigateur et le serveur. « En coulisse » veut dire : directement dans le DOM de la page du navigateur.
- La programmation **AJAX (Asynchronous JavaScript and XML)** désigne des techniques permettant des mises à jour de page web avec des échanges serveur sans recharger toute la page.
- Ces techniques s'appuient sur le **JavaScript**, le **DOM** et le protocole **HTTP** de requêtes **asynchrones**.
- AJAX permet de demander **tout type de donnée textuelle** : surtout **du XML à l'origine**, mais aussi des fichiers texte, du HTML, etc. **Aujourd'hui le format XML est remplacé par le format JSON.**

Rappels du fonctionnement d'une page WEB



Avantages

- **Diminution des temps d'attente.**
- Réduction de la **quantité de données** à envoyer et recevoir (on se limite aux données à modifier dans la page et pas toute la page).
- Création d'**applications web « single page »** (client riche) plus proche des applications natives.

Exemples

- **Google Maps :** de nouvelles sections d'une carte sont téléchargées depuis le serveur quand elles sont nécessaires sans imposer de recharger complètement la page.

  https://www.google.fr/maps/

- **Dans les sites de ventes** : avec des catalogues d'images : les changements d'image sont gérés en Ajax et non pas avec un langage serveur.

- **Draw io :** application WEB, sans rechargement de page, « single page » : https://app.diagrams.net

Asynchrone

- Un appel AJAX c'est une **requête HTTP asynchrone**.
- HTTP veut dire qu'on suit le protocole HTTP de communication entre machines (entre le client et le serveur par exemple). C'est le protocole pour demander les pages web et pour les faire transiter.
- **Synchrone** : j'attends la réponse, la page est bloquée jusqu'à l'arrivée de la réponse.
- **Asynchrone** : je continue à travailler sans attendre la réponse.

HTTP

HTTP est un protocole de communication entre machine, particulièrement entre client et serveur.

Les 2 principales requête HTTP (ou méthodes HTTP) :

- **GET : demande des ressources** au serveur. Le GET envoie de l'information pour dire ce qu'il veut et il reçoit la réponse. **Un GET ne modifie pas l'état du serveur.** Par exemple : **un download est un GET**.
- **POST : envoyer des données** au serveur pour qu'il en fasse quelque chose et qu'il nous retourne quelque chose. **Un POST modifie l'état du serveur, ne serait-ce que parce qu'il fait « tourner » un programme.** Par exemple : **un « j'aime » sur un réseau social est un POST**.

PUT est une autre méthode d'envoi de données : le POST est plutôt en création, le PUT plutôt en modification.

Interdiction du « cross-domain » : erreur CORS

- Par défaut, une requête AJAX ne peut interroger qu'un serveur situé sur son domaine : http//monsite ne peut pas interroger http://unautresite.

  ⇒ **Les requêtes « Cross Domain » sont interdites.**
- Cette limitation est faite pour des raisons de sécurité.
- Un serveur web peut paramétrer son Cross-Origine Ressource Sharing – **CORS** - pour permettre de recevoir des requêtes d'un autre site : il faut gérer ça avec prudence.

  ⇒ Un service WEB = une API (web) est un code serveur (du PHP ou du Node.js, par exemple) dont on demande l'exécution en AJAX. Ce code serveur va chercher les données sur son domaine et les retransmet ensuite au client demandeur.

Un site pour plus d'infos :

https://openclassrooms.com/courses/creez-des-pages-web-interactives-avec-javascript/interrogez-un-serveur-web

2 – AJAX Get JSON : exemples et exercices

On travaille sur le jeu de données ici :

http://bliaudet.free.fr/IMG/zip/AJAX_Get_JSON_exemples_et_exercices.zip

Le document : 0-Cours et Exercices.js présente le travail à faire/

3 - XMLHttpRequest

Présentation

On va montrer ici la technique XMLHttpRequest

Requêtes synchrones et asynchrones

Problème des requêtes synchrones

Une requête HTTP synchrone bloque la page web jusqu'à ce que la réponse soit disponible, ce qui peut prendre un certain temps.

Dans une communication avec un serveur, on prend un risque d'attente incontrôlé.

Passer en mode asynchrone : gestion d'événement

Si on passe en mode asynchrone, **la réponse devient un événement**, au même titre qu'un clic de souris.

Le traitement sera donc fait dans une méthode attachée à un Listener.

Étapes générales du code

1) Création d'une requête HTTP : new XMLHttpRequest ()
2) Écriture de la requête : méthode open ()
3) Envoi de la requête : méthode send ()
4) Création d'un listener : attribut-événement onreadystatechange
5) Écriture des traitements dans la fonction du listener (parfois appelé callback).

Exemple : GET : récupération de données JSON sur le serveur

Mais l'url est en local : on récupère les données sur notre machine : adresse http://localhost/etc.

```
request = new XMLHttpRequest();
url = "http://localhost/javascript-web-srv/data/films.json"
request.open("GET", url)
request.send(null);
```

Principes

On crée un objet XMLHttpRequest.

On appelle la méthode open() avec une GET et l'url demandée : ici un fichier JSON.

On ne passe rien en paramètre dans le send()

Explications

- ***Création d'un objet XMLHttpRequest()***

  C'est l'objet standard pour faire de l'AJAX et envoyer une requête HTTP en JavaScript.

- ***La méthode open()***

  Elle permet de définir le type de requête HTTP : ici un GET. Et aussi de définir l'URL cible : c'est un fichier JSON qui se trouve sur notre serveur. C'est le fichier qu'on va récupérer (GET).

- ***send ()***

  La méthode send() envoie la requête.

  Ici la méthode n'a pas de paramètre.

  Les paramètres serviront pour les requêtes POST.

Traitements asynchrone

```
request.onreadystatechange = function() {
  console.log("this.readyState : ", this.readyState);
  // CAS D'ERREUR
  if(this.readyState != 4) return;
  if(this.status != 200){
    alert("Erreur Ajax : aucune donnée reçue")
    return;
  }
  if(this.responseText == null){
    alert("Erreur Ajax : "+ this.statusText)
    return;
  }

  // CAS GENERAL
  console.log("this.responseText : ")
  console.log(this.responseText);

  var films = JSON.parse(this.responseText);
  var baliseInfo=document.getElementById("info");
  baliseInfo.innerHTML='Liste des films';


  films.forEach( function (film) {
    console.log(film.annee+"-"+film.titre+"-"+ film.realisateur);
    var noeud = document.createElement("li");
    noeud.textContent =
      film.annee+"-"+film.titre+"-"+ film.realisateur
    baliseInfo appendChild(noeud);
  });
}
```

Explications : gestion des erreurs

Sur l'objet « request » de la classe XMLHttpRequest, on ajoute un attribut de type gestion d'événement : « onreadystatechange » (même logique qu'un « onclick »). On crée donc un listener (un écouteur) qui va réagir à chaque réponse du serveur en exécutant la fonction qui est écrite ensuite.

Dans la fonction, on commence par traiter les cas d'erreur.

Si l'attribut « readyState » est != 4, on peut stopper l'exécution de la fonction. Etc.

Explications : gestion de l'attribut responseText

Ensuite on peut passer au cas général.

On récupère les données reçues par l'attribut « responseText ».

La principale instruction est donc : films = JSON.parse(this.responseText) => on parse les données JSON pour obtenir un tableau d'objets JavaScript.

Une fois ça fait, c'est du code JavaScript classique consistant à mettre la réponse dans un innerHTML : on parcourt le tableau de films pour ajouter des li dans la div.

Bilan

On a récupéré un fichier JSON sur le serveur, en asynchrone et on a afficher les données dans la page HTML.

Présentation détaillée de l'objet XMLHttpRequest

Création d'un objet XMLHttpRequest

L'objet XMLHttpRequest permet de gérer les échanges HTTP entre machines.

Tous les navigateurs ont adopté l'objet XMLHttpRequest.

Pour créer un objet XMLHttpRequest :

```
request = new XMLHttpRequest();
```

Cet objet va nous permettre de créer des requêtes HTTP en JavaScript.

- ***Historique***

  Il a été initialement inventé par Microsoft pour Internet Explorer en 1999 comme objet ActiveX. ActiveX est une technologie de dialogue entre programme permettant, entre autres, de gérer des mises à jour de logiciel. Les navigateurs ont gardé l'objet sans prendre toute la technologie ActiveX

- ***Compatibilité***

  Les vieilles versions d'IE ne sont pas compatibles. Il faut faire un traitement à part pour gérer ça.

- ***Documentation***

  https://developer.mozilla.org/fr/docs/Web/API/XMLHttpRequest

  https://openclassrooms.com/courses/les-requetes-http

Les méthodes

- ***open (methode, url, asychrone)***

  La méthode open permet de configurer la requête HTTP avant son lancement.

  3 paramètres :

  - **le type de requête HTTP** (le plus souvent GET, POST ou PUT (le POST plutôt en création, le PUT plutôt en modification),
  - **l'URL cible**
  - false si la requête est synchrone, true ou rien sinon.

- ***send (data)***

  La méthode send envoie la requête HTTP vers l'URL cible fournie à open.

  1 paramètre :

  - L'information envoyée au serveur, quand il y en a une : requêtes POST ou PUT
  - NULL sinon : requête GET.

- ***setRequestHeader (clé, valeur)***

  La méthode permet de définir des couple clé-valeur (champ-valeur) pour configurer un peu plus la requête en cas de POST. Il faut définir le content-type.

  Paramètres usuels pour un fichier texte : ("Content-type", "application/x-www-form-urlencoded")

Les attributs

- ***responseText***

  L'attribut responseText contient sous forme textuelle la réponse renvoyée par le serveur à la requête HTTP.

  Ca peut être du code HTML. Ou un format JSON : dans ce cas, il peut être transféré facilement dans un tableau

- ***responseXML***

  Les données sont envoyées au format XML. Elles sont alors exploitables avec le DOM.

- ***readyState***

  Indique le statut de la requête : 0=non initialisée, 1=chargement en cours, 2=chargée, 3=interactive, 4=terminée.

- ***status***

  Code de retour de serveur. Utile pour la gestion des erreurs.

  Les erreurs les plus courantes sont :

  - URL cible incorrecte
  - serveur indisponible
  - réseau indisponible

  Liste des codes erreurs : http://www.codeshttp.com

Attribut d'événement

- ***onreadystatechange***

  On peut lui affecter une fonction qui sera appelée à chaque fois que le readyState change.

Le tri

On écrit une fonction de tri pour trier le tableau des données selon une clé d'objet.

Il s'agit de pur JavaScript, pas spécialement d'AJAX :

```
function sortByAnnee(key1, key2){
   if (key1.annee > key2.annee) return 1;
   if (key1.annee < key2.annee) return -1;
   if (key1.annee == key2.annee) return 0;
}
```

Cette fonction est ensuite appliquée au tableau des films qu'on a récupéré.

```
films.sort(sortByAnnee);
```

4 - POST

Principes

La logique d'un POST, c'est de modifier d'une façon ou d'une autre l'état du serveur.

Un POST va toujours faire appel à un fichier PHP ou Node.js (par exemple) qui sera exécuté sur le serveur (et cette exécution est la modification minimum qui aura lieu). Ce fichier peut aussi modifier la BD ou les fichiers sur le serveur. Il peut aussi, comme avec un GET, renvoyer des données au client.

L'accès au PHP est asynchrone et n'est pas cross domain. Le PHP contient du code synchrone classique qui ne pose pas de problème de cross domain.

Le cœur du code AJAX sera donc le open() avec un POST :

```
request.open("POST", "fichier.php")
```

Pour passer des paramètres au fichier PHP, on suit la même logique qu'en PHP avec les formulaires : on va définir des couples key-values qui seront passés en paramètres du send() :

```
request.send("key=value");
```

Le reste est à peu près identique à ce qu'on a vu avec les GET.

Exemples 02-1 : POST : récupération d'un fichier JSON par une méthode POST

Cœur du Script JS :

```
var request = new XMLHttpRequest();

var url = "http://localhost/AJAX-php/03-POST/postURL.php"

var key="url";
var value = "http://localhost/AJAX-data/films.json"
params = key+"="+value;

request.open("POST", url)
// couple clé-valeur pour le header : pour le POST
request.setRequestHeader
     ("Content-type", "application/x-www-form-urlencoded");
request.send(params);
```

Principes

On demande l'exécution du fichier « postURL.php » en lui passant en paramètre un couple key-value : key=url, value = « http://localhost/AJAX-data/data/films.json ».

C'est un POST pour du PHP : $_POST['url'] contiendra « http://localhost/AJAX-data/data/films.json ».

Explications

- ***setRequestHeader ()***

  Permet de configurer le POST : ici quand on poste du texte (le POST, c'est ce qu'on va récupérer).

- ***send ()***

  La méthode send envoie des données à l'URL cible de l'open. Ici, c'est une URL (donc un fichier HTML qu'on envoi à notre fichier PHP). Le fichier PHP va transformer l'URL en fichier texte et renvoyer le résultat à notre page HTML.

Traitements asynchrones

Identiques à ceux fait avec une méthode GET.

Le fichier PHP

Identique à celui avec une méthode GET : on affiche le contenu du fichier :

```
echo file_get_contents(nettoieString($_POST['url']));        }
```

Bilan

C'est équivalent fonctionnellement au GET.

Tests

- On peut tester en localhost : ça fonctionne.
- On peut tester directement à partir d'un navigateur : on retrouve un problème de cross-origine : CORS. On peut régler ce problème en paramétrant le serveur Apache, mais c'est déconseillé.
- On peut tester à partir d'une version qui se trouve directement sur un serveur : ça fonctionne -> http://bliaudet.free.fr/AJAX/02-2-ajaxPostJSON-bliaudet.html

5 - POST et BD

Principes

On peut appeler n'importe quel fichier PHP : ce fichier peut donc faire des actions dans la BD, en lecture ou en écriture.

On pourra passer des paramètres si on veut spécifier la lecture ou faire une écriture.

Le principe est le même que précédemment.

La différence principale vient du fichier PHP.

Exemples 02-2 : POST : récupération des données d'une BD avec une méthode POST

Cœur du Script JS :

C'est le même que précédemment, sauf qu'on ne va pas passer de paramètres au send()

```
  var request = new XMLHttpRequest();
  var url = "http://localhost/AJAX-php/04-POST-
BD/page_oeuvres.php"
  request.open("POST", url)
  // couple clé-valeur pour le header : pour le POST
  request.setRequestHeader("Content-type", "application/x-www-
form-urlencoded");
  request.send(null);
  var url = "http://localhost/AJAX-php/03-POST/postURL.php"
```

Le fichier PHP

Le fichier PHP gère un accès à une BD dans une programmation objet. Il fait un simple affichage des données avec un echo.

Cet echo est ensuite traité par le fichier HTML.

```
echo file_get_contents(nettoieString($_POST['url']));        }
```

Traitements asynchrones

On affiche les données directement.

Bilan

On peut accéder à une BD avec AJAX. L'exemple est en lecture. On pourrait passer des paramètres pour spécifier la lecture. On pourrait aussi appeler un fichier PHP qui gère de l'écriture.

Le code PHP est simple si on met en place une architecture MVC côté PHP.

Tests

- Il faut charger la BD qui se trouve dans : AJAX-php/04-POST-BD/BD_ARTISTE.sql

  ⇒ Le fichier connexion.php (dans le même répertoire) définit le password le password de connexion pour root : mettez-le à jour si nécessaire.

- On teste en localhost : ça fonctionne.

UTILISER DES API WEB

Présentation

https://openclassrooms.com/courses/creez-des-pages-web-interactives-avec-javascript/interrogez-un-serveur-web

Résumé

- Une API est un ensemble de services offert par un logiciel à d'autres logiciels. Grâce aux API, les programmes informatiques peuvent interagir entre eux.
- Les API web sont des API accessibles via HTTP ou HTTPS. Elles utilisent souvent le format JSON. On consomme une API web dans un programme au moyen d'un appel AJAX.
- Pour consommer une API, il faut étudier sa documentation ou la tester à l'aide d'une extension de navigateur comme RESTClient ou **Postman**.
- Certaines API sont ouvertes. D'autres sont soumises à authentification, par exemple au moyen d'une clé d'accès.
- Une API REST est une API qui suit un protocole standardisé.

API web = Web Service

Aujourd'hui, on parle d'API web. Hier on parlait de « Web Services ». C'est là même chose.

Si on google « instagram api » ou « instagram api web » ou « instagram web services », on otient le même résultat.

Définition

Un nombre croissant de sites et de services en ligne proposent des **API web destinées aux développeurs**.

Une API web d'un site « S » offre aux développeurs le **moyen d'exploiter les données et les services du site « S »** pour leur propre site.

Les API sont des services « B to B »

API (Application Programming Interface ou interface de programmation)

=

Ensemble de services fournis par une application web pour d'autres applications web.

Une API va permettre :

- de récupérer la météo
- de récupérer les places libres dans un train
- de se localiser sur une carte

Plus d'info :

https://medium.com/@mercier_remi/c-est-quoi-une-api-f37ae350cb9#.4mkkuuvyd

La technologie des API Web est un moyen rapide de distribution d'information entre sites.

Consommer et produire une API

Une application peut « consommer » des API : les utiliser. C'est l'appli-conso.

Elle peut aussi produire une ou plusieurs API pour d'autres applications. C'est l'appli-prod.

Ainsi, grâce aux API, les applications peuvent interagir entre elles. Ca permet de mettre en œuvre les applications N tiers.

Précisions sur la notion d'interface

Application Programming Interface

Interface : entre moi et la télé, la télécommande est l'interface qui me permet d'utiliser la télé. La télé « produit » l'interface. J'utilise l'interface.

Au restaurant : le menu, interface entre moi et le serveur. Le serveur « produit » l'interface. J'utilise l'interface.

2 applications qui veulent se parler entre elles passent par une interface.

1 application propose une interface dont une autre se sert (la télévision propose sa télécommande, le restaurant propose un menu).

Une application utilise une interface proposée par une autre (j'utilise la télécommande de la télévision, j'utilise le menu du restaurant).

Par exemple, si je veux utiliser Paypal sur mon site, j'utiliserai l'API de Paypal.

Mon site peut utiliser les API d'autres sites et/ou proposer une API pour d'autres sites.

Une API déterminera quand et à quoi on peut accéder.

Contenu des API

- Une **bibliothèque** de **fonctions** et/ou de **classes** et/ou d'**objets** : les bibliothèques Java sont des API, le DOM JavaScript est une API.
- Des **données** et/ou des **fonctions** pour les utiliser à partir d'un site WEB
- Du code qu'on peut inclure dans son programme.

API WEB – Asynchrone - AJAX

- Une API web est une API accessible via les technologies Web, notamment les protocole HTTP ou HTTPS. Elles vont souvent s'appuyer sur des **technologie AJAX :** la requête est plutôt **asynchrone** pour éviter d'attendre la réponse si elle ne vient pas**.**
- Pour consommer (utiliser) une API web, il faut connaître son adresse (URL) et son mode de fonctionnement (souvent JSON pour les échanges de données).
- API REST : c'est un type particulier d'API. C'est un standard qui permet de normaliser le fonctionnement des API.

3 sortes d'API

- API ouvertes
- API par clé
- Iframe (déprécié) - widget

Les API web ouvertes

Présentation

On cherche des API pour enrichir nos pages web.

Une fois l'API trouvée, il faut étudier sa documentation.

Les données sont accessibles sans clé.

Exemple 1 : flickr

flickr : site de partage de photos

Sa documentation : https://www.flickr.com/services/api/

Exemple 2 : le gouvernement

https://www.data.gouv.fr/fr/faq/

https://www.data.gouv.fr/api/1/organizations/premier-ministre/

API avec authentification par clé d'accès

Présentation

La plupart des API imposent des limitations d'accès.

Identification par clé d'accès (acces key)

Chaque service web peut utiliser sa propre technique pour générer ses clés d'accès, les distribuer aux clients puis surveiller leur utilisation.

La clé d'accès se présente souvent sous la forme d'une longue série de lettres et de chiffres ajoutée dans l'URL de l'API.

Ca permet au fournisseur de l'API se suivre les accès des ses clients.

Exemple d'URL avec clé d'accès

http://api.wunderground.com/api/50a65432f17cf542/conditions/q/France/Lyon.json

Cet exemple ne fonctionne plus car la clé n'est plus valable.

iframe – widget : API HTML intégrée

Présentation

De plus en plus de site propose une API sous la forme d'une <iframe> (déprécié pour les sites en https) ou un widget dans une balise à intégrer directement dans le code HTML.

Dans ce cas, on n'a plus besoin de code JavaScript ou Ajax.

L'avantage des widget, c'est qu'elles sont plus facile à paramétrer : quand on regarde le code, on voit facilement où intervenir pour modifier des paramètres CSS.

Les iframe sont plus obscures.

Exemple widget

http://www.meteovista.fr/Widgets-Meteo/4266446/0

Intégration d'un code HTML avec API météo :

```
<div style="font-family: Arial;background-color: #fbfbfb;border:
1px solid #e7e7e7;width: 255px;height: 335px;-moz-box-shadow: 0 0
2px 1px #e7e7e7;-webkit-box-shadow: 0 0 2px 1px #e7e7e7;box-
shadow: 0 0 2px 1px #e7e7e7;overflow: hidden; -webkit-border-
radius: 4px; -moz-border-radius: 4px; border-radius: 4px;">
    <div style="width: 255px;height: 335px;">
        <div style="margin:7px 10px;">
            <div style="color: #222222;font-family: Arial;font-
size: 12px;font-weight: bold;margin: 0px 0px 7px 0px;line-height:
14px;">
                Prévisions météorologiques<br/><span style="font-
weight:normal;">Paris</span>
            </div>
            <iframe id="widget-frame"
src="http://www.meteovista.fr/Go/ExternalWidgetsNew/ThreeDaysCity
?gid=4266446&sizeType=1&temperatureScale=Celsius&defaultSettings=
True" width="235" height="216" frameborder="0" scrolling="no"
style="border: none;" allowtransparency="true"> </iframe>
            <a
href="http://www.meteovista.fr/Europe/France/Paris/4266446"
style="background:
url(http://www.meteovista.fr/Shared/Images/list_icon_blue_trans.p
ng) no-repeat scroll left 1px transparent;color: #0160b2;font-
family: Arial;font-size: 12px;font-weight: normal;padding-left:
14px;margin: 7px 0px 5px 0px;line-height: 12px;outline:
none;text-decoration: none;display: inline-block;"
target="_blank">Météo à Paris</a>
            <a href="http://www.meteovista.fr/" style="display:
block;height: 25px;width: 113px;margin: 0px 10px 8px 0px;outline:
none;text-decoration: none;" title="Meteovista.fr la météo
autrement" target="_blank"><img
src="http://www.meteovista.fr/Shared/Images/variations/fr-FR/new-
widget-logo-FR-color.png" width="113" height="25"
alt="Meteovista.fr la météo autrement" style="border:
none;background-color: transparent;box-shadow: none;" /></a>
        </div>
    </div>
</div>
```

Exemple iframe

http://www.infoclimat.fr/api-previsions-meteo.html?id=2988507&cntry=FR

Intégration d'un code HTML avec API météo :

```
<iframe seamless width="888" height="336" frameborder="0"
        src="http://www.infoclimat.fr/public-
api/mixed/iframeSLIDE?_ll=48.85341,2.3488&_inc=WyJQYXJpcyIsIjQyIi
wiMjk4ODUwNyIsIkZSIl0=&_auth=BR9RRgR6UXNUeQQzAHZReFgwAzYAdgQjBnpS
MV04XiMJYgNiDm4GYFI8Ui8OIQs9AC1UNwE6BTVTOFcvWigAYQVvUT0Eb1E2VDsEY
QAvUXpYdgNiACAEIwZkUjddOF4jCW0Dbg5zBmVSPlI1DiALPgA6VDIBIQUiUzFXNF
oxAGUFYFE3BGRRNlQ%2BBGEAL1F6WG4DZAA5BG0GMFIzXTReOAlqA2EOPgZlUjRSN
g4gCzwANFQ9ATYFOlM1VzRaMgB8BXlRTAQUUS5UewQkAGVRI1h2AzYAYQRo&_c=a3
6b378f334c1f3406bd386ba10812a8"
></iframe>
```

API REST

https://openclassrooms.com/courses/utilisez-des-api-rest-dans-vos-projets-web

Présentation

REST est un protocole créé en 2000.

Une API REST suit le **protocole REST.**

Comme toute API, une API REST est basée sur HTTP (HyperText Transfert Protocole). Le client peut exécuter des méthodes GET, PUT, POST, DELETE, etc.

Critères d'une API REST :

- **sans état** (pas d'info conservé sur le client par le serveur entre deux requêtes, pas d'historique),
- **avec cache** (le client peut garder les informations en cache pour éviter de recharger une requête),
- orienté client serveur,
- interface uniforme,
- système de couches,
- code à la demande (optionnel).

Format des réponses du serveur pour les API REST : **souvent JSON**, XML, CSV, ou même RSS.

Une API REST conforme aux standards est dite RESTful.

Alternative : SOAP

SOAP : Simple Object Access Protocol.

SOAP est un protocole d'accès à des services et des données.

SOAP définit une méthode de communication avec des règles strictes.

SOAP a été défini par Microsoft et IBM en remplacement de CORBA et DCOM. C'est devenu une norme du W3C.

C'est plus sécurisé : pour les banques !

Les données sont souvent renvoyées au **format XML**.

SOAP utilise le WSDL (Web Service Description Langage) : c'est plus compliqué à coder que REST.

Le protocole REST est une alternative au protocole SOAP : REST est plus simple à utiliser.

Les API REST sont les plus nombreuses, largement.

Quelques exemples d'API REST

Instagram API

Googler instagram api

Documentation : https://www.instagram.com/developer/

Je vais pouvoir faire de mon application un utilisateur d'instagram : elle peut récupérer tout ce qui se trouve sur instagram.

L'onglet « authetification » va permettre de recevoir un « ACCESS_TOCKEN »

L'onglet « Endpoints » permet, avec un ACCESS_TOCKEN, on pourra récupérer des photos.

Gmail API

Googler gmail api

Avec l'api gmail, j'ai accès aux fonctionnalités d'envoi de mails. J'accède aux messages, brouillons, dossiers, etc. de l'utilisateur gmail qui sera connecté avec ses codes.

Github API

Googler github api

On peut héberger son code sur github. Avec l'api github, on peut récupérer les statistiques par exemple, sur mon application. On peut par exemple notifier les utilisateurs du fait qu'il se passe quelque chose.

Etc. !!!

Notion de ressource

Présentation

Une ressource c'est quelque chose que quelqu'un va chercher.

Sur instagram, une ressource c'est une photo ou une liste de photos.

Sur gmail, c'est un message.

Une API organise les ressources pour qu'elles soient faciles à récupérer par les utilisateurs.

Le format est celui de l'URI : Uniform Ressource Identifier (URL : Uniform Ressource Locator).

L'URI définit donc une sorte de syntaxe standard pour faciliter l'écriture du code.

Usage des URI

/users : les utilisateurs

/users/238 : l'utilisateur 238

/users/238/adresses : les adresses de l'utilisateur 238

/users/238/adresses/195/notes : les notes de l'adresse 195 de l'utilisateur 238

Cette dernière URI est trop complexe. On utilisera :

/adresses/195/notes : cela sous-entend que c'est pour un utilisateur donné