Technologies du web

6 - AJAX

Types de contenus

Page statique : le contenu est défini à l'avance et figé dans un fichier servi directement
→ Lecture (page perso des années 90)

Page dynamique : le visiteur peut modifier une partie ou toute la page
→ Lecture, écriture (blog, wiki, forum, etc.)

Application en une page : le contenu est plus complexe
→ Manipulation de ressources (webmail, gestionnaire de fichiers, chat, etc.)

Services Web

Ressource web conçue pour des robots

Ex : https://api.tela-botanica.org/service:del:0.1/protocoles


```
{
  "entete": {
    "masque": "",
    "total": 5,
    "depart": 0,
    "limite": 10
  },
  "resultats": {
    "1": {
      "protocole.id": "1",
      "protocole.intitule": "Aide à l'identification",
      "protocole.descriptif": "Choisissez les photos les plus utiles pour vérifier la détermination d'une espèce",
      "protocole.tag": "caractere",
      "protocole.mots_cles": "",
      "protocole.identifie": "0"
    },
    "3": {
      "protocole.id": "3",
      "protocole.intitule": "Capitalisation d'images",
      "protocole.descriptif": "photographier en extérieur les organes (feuille, fruit, tronc, etc.) de plantes et transmettre
      "protocole.tag": "Plantnet",
      "protocole.mots_cles": "port,fleur,fruit,feuille,plantscan_new,ecorce,rameau,planche",
      "protocole.identifie": "0"
```

Services Web

Dans l’exemple précédent on accède aux informations de différents protocoles, on peut lire leur nom et leurs caractéristiques. Ces informations seront lues par l’application puis traitées et/ou affichées.

Ce service web permet de faire le lien entre l’application backend et les applications frontend.

Autres usages notables :

- Automatiser la collecte et la modification de données
- Pour échanger des données entre applications distinctes
- Se connecter avec des identifiants d’un autre site (OAuth)

API

Application Programming Interface

Ensemble de services web permettant de manipuler des ressources : les consulter, les modifier, etc.

REST

Architecture de service web utilisant des propriétés strictes :

Une ressource (une collection ou un élément) identifiée via une **URI**

Appelée par un verbe **HTTP** (GET, POST, PUT, DELETE et OPTIONS)

Accompagnée des liens décrivant les autres actions possibles (HATEOAS)

À noter que GET ne modifie par la ressource, et que PUT et DELETE sont idempotents (répéter la même requête donnera le même résultat)

Appeler un service REST

On dit qu’on peut appeler un service, l’utiliser, le consommer :

- depuis un programme serveur, en PHP
- depuis un programme client, avec AJAX (en JavaScript dans le navigateur)

En PHP

```
// On va utiliser CURL via l'extension PHP dédiée
$url = 'https://api.tela-botanica.org/service:del:0.1/protocoles';

$curl = curl_init();
curl_setopt($curl, CURLOPT_URL, $url);
// CURLOPT_RETURNTRANSFER permet de récupérer le contenu de la page
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

$page = curl_exec($curl);

curl_close($curl);

$contenu = json_decode($page);
```

Retourner un code HTTP

Les API doivent retourner un code HTTP en fonction du résultat.

Les codes les plus courants sont :

- 200 : succès de la requête ;
- 301 et 302 : redirection, respectivement permanente et temporaire ;
- 401 : utilisateur non authentifié ;
- 403 : accès refusé ;
- 404 : page non trouvée ;
- 500 et 503 : erreur serveur ;
- 504 : le serveur n'a pas répondu.

https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP

Retourner un code HTTP

La plupart du temps un code 200 est adapté, il indique que votre API a bien compris la requête.

Si le résultat n’est pas celui attendu il convient d’ajouter un message d’erreur expliquant le problème avec la requête, par exemple si le jeu de résultat est vide ou si le format des paramètres n’est pas adapté, ou qu’un paramètre est manquant.

Retourner un message approprié

```php
<?php

$response = [];

if (!isset($_REQUEST['category']) {

  $response["error"] = "Paramètre 'category' manquant");

}

// ...

echo json_encode($response);
```

En JavaScript dans le navigateur

On va utiliser AJAX (Asynchronous JavaScript and XML)

(XML était le format utilisé historiquement, mais on va plutôt utiliser JSON)

Ces fonctionnalités permettant d'appeler des services web sont rassemblées dans l'objet **XMLHttpRequest**

Asynchronous

AJAX c’est surtout le côté **asynchrone** :

L’appel effectué au service web par AJAX ne bloque pas le reste de l’exécution du script. **La page n’est pas bloquée en attendant la réponse du service.**

Lorsque la réponse viendra c’est une fonction particulière qui s’occupera de la traiter : cette fonction c’est un **callback**

Interlude

Pour résumer, on a deux grands types de pages :

Les pages statiques sans AJAX

Les pages dynamiques avec AJAX

Sans AJAX

Une nouvelle page est envoyée entièrement à chaque fois au navigateur :

Si on clique sur un lien

Si on valide un formulaire

Etc.

Avec AJAX

On peut **modifier partiellement le contenu de la page**, recharger une image, changer un texte, ajouter des éléments, **sans recharger entièrement la page**.

Un appel AJAX est envoyé au clic sur un lien ou lors de la validation d'un formulaire
Une requête asynchrone est émise, et lorsque la réponse arrive une fonction se charge de l'interpréter et d'ajuster le contenu de la page courage en fonction.

À quoi ça sert ?

Économiser en bande passante et en temps de traitement coté serveur, ne recharger que ce qui est nécessaire.

Améliorer la réactivité de la page.

Garder une page à jour en continu, les nouvelles informations peuvent arriver au fur et à mesure en quasi temps réel.

AJAX : XMLHttpRequest

Exemple de requête AJAX avec l'objet **XMLHttpRequest** :

```
var ajax = new XMLHttpRequest();

ajax.addEventListener('load', function(data) {
    console.log(data.target.responseText);
});
ajax.addEventListener('error', function(data) {
    console.log('Oups, erreur : ', data);
});

ajax.open('GET', 'http://rorto.fr/api/news.php');
ajax.send();
```

Exemple

Une page avec rafraichissement automatique toutes les 10 secondes chargeant un flux d’actualités.

La page HTML exécutera un appel AJAX toutes les 10 secondes

Le service appelé renverra les actualités dans du JSON

La page sera modifiée en fonction de la réponse

Service web

L’appel au service renverra du JSON :

curl http://localhost:8080/api/news.php

{ "news": [{ "title": "news1", "content": "text" }, { "title": "news2", ... }] }

Service web en PHP

```php
<?php
$news = [
 "breaking" => [
      [ "title" => "news1", "content" => "text" ],
      [ "title" => "news2", "content" => "etc..." ],
 ],
 "news" => [
   [ "title" => "breaking", "content" => "hop" ],
 ]
];

if (isset($_REQUEST['category'])) {
 $category = $_REQUEST['category'];
 if (!array_key_exists($category, $news)) {
   http_response_code(204);
   exit();
 }
 echo json_encode($news[$category]);
} else {
 echo json_encode($news);
}
```

Structure de la page HTML

```
<body>
    …
    <section id="news">
    </section>
    …
</body>
```

Code JavaScript

```
document.addEventListener('DOMContentLoaded', function () {
    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('#news).innerHTML = new_html;
        });

        request.open("GET", "api/news.php");
        request.send();
    }

    // rafraîchissement toutes les 10 secondes
    windows.setInterval(refresh_news, 10000);
});
```

Restrictions

Pour des raisons de sécurité **un service web ne pourra être appelé que par une page appartenant au même domaine**. C'est le "same-origin policy".

Si on prend l'exemple d'une application sur
http://store.company.com/dir/page.html

http://store.company.com/dir2/other.html	Succès
http://store.company.com/dir/inner/another.html	Succès
https://store.company.com/secure.html	Échec (Protocoles différents)
http://store.company.com**:81**/dir/etc.html	Échec (Port différents)
http://**news.**company.com/dir/other.html	Échec (Hôtes différents)

Mémo

Un service web est interrogé par un programme

Un service web REST est interrogé via une URL et en HTTP

AJAX dynamise le contenu des pages HTML

Un service web peut être écrit en PHP sur le serveur

Conclusion

Les webservices sont largement utilisés dans les applications web

Ces applications sont dynamiques et interactives

Les services web permettent la création d’architecture distribuées

Les services web font le lien entre l’interface utilisateur (frontend) et la partie métier qui gère les données (backend)

Ressources

Les bases en vidéo et en français :

HTML : https://www.grafikart.fr/formations/html
CSS : https://www.grafikart.fr/formations/css
PHP : https://www.grafikart.fr/formations/php-debutant
JS : https://www.grafikart.fr/formations/debuter-javascript
Serveur : https://www.grafikart.fr/formations/serveur-linux

Parties 1, 2 et 3 :
https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql