INF3005 – Programmation web avancée

Ajax

Jacques Berger

Objectifs

Introduire le paradigme Ajax

Prérequis

Flask

HTML

Javascript

Javascript

Les applications web utilisent beaucoup le Javascript pour les fonctionnalités du frontend

Avec Flask, placez les scripts JS dans le répertoire static

Inclure les scripts JS dans les templates

Javascript

Le Javascript peut servir à plusieurs choses :

- Afficher/cacher des éléments
- Valider un formulaire
- Envoyer des requêtes Ajax

Javascript

Les validations faites dans le frontend doivent toutes être refaites dans le backend, car tous les éléments du frontend peuvent être contournés

Voir exemple

Ajax

Ce n'est pas un langage, ni une technologie, c'est un modèle de communication entre le fureteur et le serveur web

Originalement : Asynchronous Javascript And Xml
Maintenant un nom commun

Ajax

Modèles

- Application web traditionnelle
- Application Ajax

Ajax

Application traditionnelle

Un appel au serveur pour chaque modification de la page

À chaque fois, le serveur génère la page au complet et le fureteur l'interprète au complet

Full Page Refresh

Ajax

Application Ajax

Lors du chargement initial, le serveur envoie la page web et l'application Javascript

Ensuite, le Javascript s'occupe de communiquer avec le serveur de façon asynchrone

Le Javascript met à jour la page dynamiquement lorsque le serveur répond

Ajax

Idées de base

- Éviter le full page refresh
- Améliorer la convivialité (fluidité) de la page web
- Diminuer la charge de travail du serveur web

Ajax

Les communications subséquentes entre le fureteur et le serveur peuvent être sérialisées de plusieurs façons :

- XML
- HTML
- JSON
- Données brutes

Ajax

Données brutes
- Peu de bande passante
- Peu de travail pour le serveur
- Facile à manipuler avec le Javascript
- À éviter pour les chaînes de caractères
  - Encodage non spécifié

Ajax

HTML

- Le serveur ne génère qu'un fragment HTML
- Ce fragment sera inséré au bon endroit
- Facile à manipuler avec Javascript
- Plus volumineux que les données brutes
- Méthode très populaire

Ajax

XML
- Volumineux
- Plus complexe à générer
- Facile à manipuler
- Méthode favorisée dans les débuts d'Ajax
- Recommandée pour les données avec encodage de caractères

Ajax

JSON
- Moins volumineux que XML
- Tend à remplacer XML
- Méthode populaire actuellement

Ajax

Fonctionnement

On doit concevoir notre application du côté du serveur de façon à recevoir une requête HTTP et générer du contenu sérialisé (HTML, XML, JSON, brut)

Ajax

Fonctionnement

Le client doit communiquer avec la fonctionnalité Ajax du serveur

Le Javascript utilisera l'objet XMLHttpRequest

XMLHttpRequest

Objet servant à :

- Envoyer une requête HTTP
- Traiter la réponse HTTP

XMLHttpRequest

Exemple synchrone

```
var requestObject = new XMLHttpRequest();
requestObject.open("GET", "/liste", false);
requestObject.send();
var resultat = requestObject.responseText;
```

XMLHttpRequest

Méthode open

Paramètre #1 : méthode HTTP

La méthode HTTP à utiliser pour envoyer la requête, habituellement GET ou POST

XMLHttpRequest

Méthode open

Paramètre #2 : URL

L'URL de la requête (relative ou absolue)

Paramètre #3 : asynchrone

true pour envoyer la requête de façon asynchrone (ajax), false pour l'envoyer de façon synchrone

XMLHttpRequest

Méthode send
- Envoie la requête HTTP

Paramètre optionnel : Les données à envoyer au serveur si POST
- Données brutes
- XML
- autre

XMLHttpRequest

Propriété responseText
    Le résultat textuel de la requête

Propriété status
    Le statut de la réponse

Ajax

La requête doit être envoyée au serveur de façon asynchrone pour éviter de bloquer l'utilisateur dans ses actions

Au préalable, on a enregistré une méthode à appeler lorsque le XMLHttpRequest recevra une réponse du serveur

Ajax

Enregistrement de la méthode et envoie de la requête

Propriété onreadystatechange

```
var request = new XMLHttpRequest();
request.open("GET", url, true);
request.onreadystatechange = updatePage;
request.send();
```

Ajax

La méthode va être invoquée à chaque changement d'état de l'objet XMLHttpRequest

On doit obtenir le numéro de l'état qui nous intéresse

Ajax

0 : Requête non initialisée
1 : Connexion établie avec le serveur
2 : Requête reçue par le serveur
3 : Le serveur traite la requête
4 : La réponse est prête (DONE)

Ajax

On doit vérifier l'état dans la fonction appelée

Propriété readyState

```
function updatePage() {
    if (request.readyState === XMLHttpRequest.DONE) {
        // code qui vérifie le statut et traite le résultat
    }
}
```

Backend

Souvent, nous allons déclarer des routes spécifiquement pour les appels Ajax

Attention : les routes sont visibles aux utilisateurs qui inspectent le code source de la page

Fetch

Nouvelle fonctionnalité expérimentale dans les fureteurs : fetch

Appel Ajax simplifié utilisant les nouveautés du Javascript

Un peu plus loin...

XMLHttpRequest
    https://www.w3.org/TR/XMLHttpRequest/

Fetch
    https://github.github.io/fetch/