



Drupal, Utiliser Ajax pour générer du contenu

Le 25 novembre 2007 à 22:27.

Dernièrement je faisais quelques recherches sur Drupal et [AJAX](#)^[1] pour me rendre compte qu'il y avait finalement assez peu de ressources, même en anglais, traitant de ce sujet. Pourtant cette technique, au delà du buzz généré autour d'elle, peut rendre de réels services et grandement fluidifier des interfaces trop lourds ou trop peu intelligent. Et pour simplifier son utilisation, rien de mieux que JQuery, inclus dans le coeur de Drupal depuis sa version 5.

Mise en oeuvre

L'exemple que je vais prendre ici est très simple et reprends l'exemple Ajax de l'horloge développé [tour ici](#)^[2]. Il s'agit simplement d'afficher une page Drupal, contenant une horloge dont l'heure est donnée, toute les secondes, par une requête Ajax.

Tout le jeu ici, est de s'intégrer parfaitement dans un module Drupal. Et ceci tant du point de vue de la page "horloge" (facile 😊 que du serveur de requête Ajax (un peu moins facile..). Cela nous permet de faire des modules totalement autonomes utilisant Ajax, mais aussi de bénéficier de toute l'infrastructure de Drupal (base de données, thèmes, etc) au niveau du serveur Ajax. Et pour réussir cette intégration, nous allons simplement utiliser le système des `callbacks`, c'est à dire des menus Drupal redirigés sur des fonctions.

La première chose à faire est donc de créer un module de base (Si vous ne savez par où commencer, vous pouvez prendre [cet exemple](#)^[3] pour démarrer). Module que nous allons nommer `exemple_ajax` et activer dans Drupal.

Ceci fait, nous allons implémenter le hook `_menu` pour définir deux menus/callback. Le premier servira à afficher la page que l'on va ajaxifier (`exemple_ajax/horloge`) et le second va être notre source de donnée AJAX (`exemple_ajax/getTime`)

```
function [4] exemple_ajax_menu($may_cache) {
  $items= array [5] ();
  if [6] ($may_cache) {} else [7] {
    $path= drupal_get_path('module', 'exemple_ajax');
    drupal_add_js($path . './exemple_ajax.js');

    $items[]= array [5] (
      'title' => 'Exemple AJAX',
      'path' => 'exemple_ajax/horloge',
      'callback' => 'exemple_ajax_horloge',
      'access' => TRUE [8],
    );

    $items[]= array [5] (
      'path' => 'exemple_ajax/getTime',
      'callback' => 'exemple_ajax_getTime',
      'access' => TRUE [8]
    );
  }
}
```

Dans le code du hook, rien de particulier, si ce n'est la présence de la fonction `drupal_add_js(...)` à qui va demander à Drupal d'ajouter à la page générée, un lien vers un fichier javascript que nous allons bientôt créer (`exemple_ajax.js`).

Ensuite, nous codons la première callback qui est chargé de l'affichage de la vue, une simple description.

```
function [4] exemple_ajax_horloge()
{
  return [9] "Bienvenue sur l'Horloge AJAX/Drupal. Il est actuellement <span
id="heure"/>";
}
```

Étape suivante, créer le fichier javascript qui va prendre en charge la mise à jour de l'horloge. Comme indiqué dans le hook menu, ceci doit être stocké dans un fichier `exemple_ajax.js` au même niveau que le module :

```
function changeHeure() {
  jquery.get("getTime.php", {}, reponseServeur);
}

function reponseServeur(texte, erreur, requete) {
  $('#heure').html(texte);
  setTimeout(changeHeure, 1000);
}
```

```
function initializePage() {
    changeHeure();
}
$(initializePage);
```

Je ne reviens pas sur ce code, il est décrit dans le [tutorial sur jQuery](#) [2].

Dernière étape, retour à Drupal et écriture de deuxième callback, celle qui va répondre à la requête AJAX :

```
function [4] exemple_ajax_getTime($argument) {
    header [10]("Cache-Control: no-cache, must-revalidate");
    header [10]("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
    print [11] date [12]("H:i:s");
    exit [13] ();
}
```

Les deux définitions de header permettent d'éviter que le navigateur ne mette en cache notre résultat de requête. Suit un simple print suivi du formatage de l'heure. La fonction termine sur un exit() pour arrêter le traitement ici. Et c'est presque là le point le plus important. Car en cassant ainsi le traitement de Drupal que l'on bénéficie de toute son infrastructure tout en ayant un contrôle parfait sur ce qui est renvoyé au client (pas de thèmes, de pages, etc..).

Vous pouvez tester la fonction dans un navigateur en y tapant l'url `exemple_ajax/getTime` qui devrait vous donner un message : `XX:YY:ZZ`.

Et voilà, c'est terminé. Il vous suffit maintenant d'aller sur l'url `/exemple_ajax/horloge` pour que la page initiale s'affiche, que le script soit chargé et modifie le comportement du message et pour que l'horloge se mette à tourner en utilisant des requêtes en Ajax, sur une source de donnée Drupal, pour se mettre à jour.

Conclusion

Une fois que l'on a les bases Ajax et jQuery, la mise en oeuvre dans Drupal est vraiment très simple et permet tout les types d'utilisation de requêtes asynchrones sur notre CMS préféré.

<http://artisan.karma-lab.net/node/1273>
(C) artisan numerique - CC BY-SA

Liens:

- [1] <http://fr.wikipedia.org/wiki/AJAX>
- [2] <http://artisan.karma-lab.net/node/1275>
- [3] <http://artisan.karma-lab.net/node/1245>
- [4] <http://www.php.net/function>
- [5] <http://www.php.net/array>
- [6] <http://www.php.net/if>
- [7] <http://www.php.net/else>
- [8] <http://www.php.net/true>
- [9] <http://www.php.net/return>
- [10] <http://www.php.net/header>
- [11] <http://www.php.net/print>
- [12] <http://www.php.net/date>
- [13] <http://www.php.net/exit>