



Drupal, créer un champ en auto-complétion

Le 14 novembre 2007 à 18:44.

Depuis Drupal 5, il est possible très simplement d'ajouter l'auto-complétion à un champ texte en utilisant une méthode dit d'appel asynchrone (aka Ajax). Traduit en français, cette fonctionnalité permet lorsque vos utilisateurs saisissent les premières lettres, de lui fournir automatique une liste de suggestion commençant par ces lettres, lui évitant de taper la suite.

Auto-Complétion

L'auto-complétion (ou suggestion) est donc un appel asynchrone. C'est à dire que lorsque l'utilisateur saisi une lettre dans le champ texte, au bout d'un temps court, une procédure JavaScript va lancer une requête prenant en paramètre le texte qui a été tapé. Le serveur (Drupal) va recevoir cette URL, et exécuter la procédure associée qui a pour tâche de construire une liste de suggestion. Ceci fait le serveur renvoi à la procédure javascript le résultat qui va l'utiliser pour fabriquer, à la volée, une petite liste déroulant pour l'utilisateur.

Dans l'exemple qui suit, le champ que nous allons modifier, permet d'obtenir une liste de termes (taxonomie), suggérés à partir de ce que l'utilisateur a saisi, et ceci pour un vocabulaire donné. Il peut être adapté à d'autre exemples facilement.

Implémenter le hook `_menu`

La première chose que nous avons à faire est donc de déclarer dans Drupal un chemin (path) qui va rediriger cette URL vers une procédure capable de lister les bonnes réponses. Et qui dit path, dit dit un hook `_menu` :

```
/**
 * Implementation of hook_menu().
 */
function [1] mon_module_menu($may_cache)
{
  $items = array [2] ();
  if [3] ($may_cache)
  {
    $items[] = array [2] (
      'path' => 'mon_module/categories/autocomplete',
      'title' => t('Auto completion pour mon module/categories'),
      'callback' => 'mon_module_categories_auto_complete',
      'access' => TRUE [4],
      'type' => MENU_CALLBACK);
  }
  return [5] $items;
}
```

Pour des raisons de performances, Drupal stocke en base de données (table `cache_menu`) tous ces chemins (c'est la raison d'être du `$may_cache`). Du coup, il est préférable après avoir écrit le hook `_menu`, de donner dans votre console de base de données un petit coup de :

```
|| DELETE FROM cache_menu;
```

Implémenter le callback

Étape suivante, écrire la procédure déclarée dans le hook `_menu` par le paramètre `callback` et qui va générer notre liste de suggestions.

```
function [1] mon_module_categories_autocomplete($string = '')
{
  $matches = array [2] ();
  $vid =
  if [3] ($string)
  {
    $cursor = db_query_range("
      SELECT name
      FROM {term_data}
      WHERE vid=10 AND LOWER(name) LIKE LOWER('%s%')", $string, 0, 10);
    while [6] ($term = db_fetch_object($cursor))
      $matches[$term->name] = $term->name;
  }
}
```

```

}
print [7] drupal_to_js($matches);
exit [8]();
}

```

Quelques explications s'imposent. Tout d'abord le paramètre `$string`. Ce dernier contient, vous l'aurez deviné, la chaîne à rechercher. Cela correspond à tout ce qui se trouve après le `mon_module/categories/autocomplete` dans l'url. Cela marche très bien sauf dans un cas, si vous voulez chercher des choses qui contiennent des `/`. Là ça marche beaucoup moins bien car `$string` ne va contenir *que* ce qui se trouve avant le premier `/` (logique). Du coup, il faut tricher un peu et rajouter une horreur de ce genre au début de la procédure :

```

|$string=substr [9]($_GET['q'], str_len('mon_module/categories/autocomplete/'));

```

Cela va prendre à la source (`$_GET['q']`) la totalité de l'URL drupal et enlever la partie qui ne sert à rien (path).

Une fois la variable `$string` en main, il suffit de faire une requête SQL pour récupérer une liste limitée à 10 items de réponse qui concordent avec notre chaîne. Notez que dans la requête d'exemple, la recherche se fait sur le vocabulaire ayant comme vid la valeur 10. Pensez à changer cela pour utiliser votre propre vid.

Une fois la liste constituée (`$matches`), on utilise la fonction magique de drupal `drupal_to_js` qui va créer une petite procédure javascript contenant nos données, utilisable par l'auto-complétion de Drupal.

Avant d'aller plus loin, nous pouvons déjà tester si tout les suggestions fonctionnent, et dans FireFox, rentrer une URL artificiel du style `http://localhost/mon_drupal?q=mon_module/categories/autocomplete/a`. Une fois exécutée, la page affichée doit contenir tout les termes qui commencent par la lettre `a`.

Ajouter le champ

Maintenant que l'on sait que notre procédure fonctionne, il ne nous reste plus qu'à ajouter dans un formulaire (soit un `mon_module_form_alter`, soit un `mon_module_form`), un champ texte tout con :

```

$form['categorie']= array [2] (
  '#type' => 'textfield', //
  '#title' => t('Tapez les premières lettres de la catégorie et laissez-vous guider...'), //
  '#required' => TRUE [4], //
  '#autocomplete_path' => 'mon_module/categories/autocomplete');

```

Rien de bien extraordinaire mis à part le champ `#autocomplete_path` qui est seul responsable de la magie. Il suffit d'afficher votre formulaire pour voir apparaître votre champ texte doté d'un étrange petit cercle gros sur la droite. Ce symbole indique que l'auto-complétion est disponible sur ce champ. Tapez une lettre (a) et attendez une seconde, une liste devrait alors apparaître avec vos termes de vocabulaire.

Conclusion

Voilà comment en peu de ligne vous pouvez changer la vie de vos utilisateurs, leur évitant les anti-ergonomiques liste déroulantes contenant des centaines d'item.

<http://artisan.karma-lab.net/node/1259>
(C) artisan numerique - CC BY-SA

Liens:

- [1] <http://www.php.net/function>
- [2] <http://www.php.net/array>
- [3] <http://www.php.net/if>
- [4] <http://www.php.net/true>
- [5] <http://www.php.net/return>
- [6] <http://www.php.net/while>
- [7] <http://www.php.net/print>
- [8] <http://www.php.net/exit>
- [9] <http://www.php.net/substr>