

Wednesday, 23 March 2005

Article sur patForms dans le International PHP Magazine

Hier j'ai reçu la confirmation que mon article sur patForms (en) va paraître dans le prochain numéro (03.05) du International PHP Magazine (en). Pour ceux qui ne connaissent pas encore patForms, c'est une librairie qui permet de générer des formulaires. Ce qui la démarque des autres, c'est avant tout sa simplicité et le fait qu'on a le contrôle total de la mise en page et du visuel du formulaire. patForms est un projet de l'équipe de PHP Application Tools (en), développé par Stéphane (en), gERD (en|de) et moi-même.

Mon article est scindé en deux parties, et la première partie qui va sortir bientôt présente l'architecture et les possibilités de base façon tutoriel avec beaucoup d'exemples. Dans la deuxième partie, j'insisterai sur la gestion des évènements, les observateurs, filtres et autres. Pour découvrir patForms c'est l'idéal, surtout que la seule source d'informations à l'heure actuelle est la collection d'exemples (en), qui est très fournie et bien documentée.

Et pourquoi encore une librairie de gestion de formulaires, me direz-vous - il est vrai qu'il y en a beaucoup, mais j'en avais marre de me battre avec des solutions jamais optimales, et je trouve que même des outils très plébiscités comme PEAR::QuickForm sont trop hermétiques et pas assez flexibles. Pourquoi? Parce-que je veux que ce soit intuitif. Si il faut que je tape 60+ de lignes de code super-cryptique juste pour afficher trois champs de formulaire, non merci. Et c'est là la vraie force de patForms: c'est super simple à utiliser - voilà tout ce qu'il faut pour créer un formulaire qui se valide tout seul:

```
// inclure les classes nécessaires
require_once 'pat/patForms.php';
require_once 'pat/patForms/Parser.php';
require_once 'pat/patErrorManager.php';

// définir les options essentielles
patForms_Parser::setNamespace( "patForms" );
patForms_Parser::setCacheDir( "cache" );

// créer l'objet formulaire à partir d'un template
$form =& patForms_Parser::createFormFromTemplate(
    'SimpleRenderer',
    'templates/form.fhtml',
    'templates/form.html'
);

// dire au formulaire de se valider tout seul
$form->setAutoValidate( 'save' );

// afficher le formulaire
echo $form->renderForm();

// le formulaire a-t-il été envoyé?
if( $form->isSubmitted() ) {
    // des erreurs de validation?
    $errors = $form->getValidationErrors();
    if( $errors ) {
        // afficher les erreurs
    } else {
        // faire qq-chose avec les données
        $values = $form->getValues();
    }
}
```

L'intéressant ici, c'est surtout le template de formulaire. Le parseur de patForms permet de générer un objet patForms à partir d'un fichier html ou encore un template patTemplate. Dans ce cas-ci, c'est un fichier html qui pourrait avoir le contenu suivant:

Faut être logué...

Pseudo:

Passe:

On peut donc écrire ses formulaires comme on le faisait avec un formulaire traditionnel, et ainsi avoir le contrôle complet du visuel. Bien sûr patForms offre aussi des méthodes simples pour créer des formulaires de manière dynamique, le parseur n'est qu'un addon.

Allez hop, j'en retourne à mes occupations mondaines - plus précisément la préparation des crêpes au sarrazin

Posté par Argh dans PHP à 17:36

Sunday, 6 March 2005

Publication de patConfiguration v2.0.0b1

Traduction du billet 'patConfiguration 2.0.0b1 released' de php-tools:

Stéphane a publié la première beta publique de son lecteur de configurations patConfiguration. À partir de cette version, la classe est basée sur des drivers avec une architecture unifiée pour lire les configurations à partir de fichiers XML, INI et WDDX avec la même interface.

Le driver XML est le plus puissant, puisqu'il permet de définir comment les balises doivent être traitées. Il est ainsi par ex. possible de définir une balise pour qu'elle soit automatiquement convertie en tableau avec les attributs comme clés et valeurs. Bien sûr on peut aussi choisir pour chaque attribut quel type de variable y est stocké pour que les valeurs soient automatiquement convertis en valeurs booléennes, nombres, chaînes de caractère et même des objets.

Pour cela, patConfiguration propose une balise spéciale très simple:

Maintenant que les balises sont définies et que patConfiguration sait comment les intégrer, on peut les utiliser dans la configuration XML:

```
getConfigValue('articles');  
?>
```

Si vous êtes inquiet que le temps que de parser des fichiers XML à chaque requête ralentit votre application, vous pouvez activer le système de cache qui rend le chargement encore plus rapide que de stocker la configuration dans un fichier PHP. Si vous voulez utiliser les définitions des balises dans plusieurs fichiers, vous pouvez utiliser des entités externes ou des balises xInclude - patConfiguration gère les deux, même sous PHP4.

Après chargement avec patConfiguration, voilà le tableau résultant de notre exemple:

```
Array  
(  
  [0] => Array  
  (  
    [vendeur] => Argh  
    [titre] => Hache Médiévale  
    [prix] => 99.99  
  )  
  
  [1] => Array  
  (  
    [vendeur] => Merlin  
    [titre] => Oeuf de dragon  
    [prix] => 500  
  )  
)
```

))
)

patConfiguration permet en outre d'accéder à la configuration complète, ou juste des parties en utilisant une mixture de la syntaxe d'accès aux tableaux dans PHP et de syntaxe de chemins à la JavaScript (boutique.articles[0], par ex.) Pour voir la liste complète des possibilités qu'offre patConfiguration, il y a les exemples en ligne ou vous pouvez télécharger le package entier par notre site.

Si vous utilisez déjà PEAT 1.4.0, vous pouvez aussi l'installer par notre serveur de canaux:

```
$ pear channel-discover pear.php-tools.net
```

```
$ pear install --alldeps pat/patConfiguration
```

En cas de bogue, utilisez notre patBugzilla.

Posté par Argh dans php-tools à 11:54

Thursday, 3 March 2005

Des générations de webdesigners se sont cassés les dents là-dessus...

Il y a un problème qui existe depuis des millénaires si on le place dans le bon contexte: les technologies Internet. Sur une échelle de temps de quatre ans, les possibilités de création de sites ainsi que les moyens et outils pour les créer ont évolué de manière exponentielle. On a finalement vu la disparition de Netscape 4, ce qui a ouvert la voie vers un HTML virtuellement sans tableaux. À partir d'Internet Explorer v5, la partie des standards qu'il gère précipite l'avènement du XHTML et des designs 'CSS Only'. L'arrivée en trombe de Mozilla et plus récemment de Firefox est encore une petite révolution à part - mais il est des choses dont on ne se débarrasse pas si facilement.

Depuis que les calques existent, et que les webdesigners utilisent cette technique pour créer menus dynamiques, fenêtres de dialogue et autres, ils se sont invariablement trouvés confrontés au champ de formulaire qui n'en a rien à cirer et qui flotte tranquillement dans son propre calque au-dessus de tout (en). C'est tout de même bizarre, non? EN quatre ans, avec tous les progrès qui ont été faits, ils n'ont même pas été capable de s'en occuper une fois pour toute. Surtout que le problème semble se concentrer sur le champ de formulaire - et indépendamment du système d'exploitation. Je comprendrais si c'était lié à un système d'exploitation spécifique (par ex. les contraintes de développement sous windows), mais là ça frise le ridicule. À moins que la technique de superimposition des champs de formulaire est commune à Windows et OSX

C'est d'autant plus ridicule si on considère que Firefox a maintenant réussi apparemment sans grand effort à nous débarrasser de ce problème. Je ne m'en suis rendu compte qu'après avoir testé les nouvelles fenêtres de dialogue dans l'interface d'administration de Syrocco avec Internet Explorer: en fait, j'avais complètement oublié le problème. Il ne reste plus qu'à espérer qu'Internet Explorer 7 ira dans la même direction...

Posté par Argh dans Web à 11:55