Classe SimpleListingHelper

V1.1.1

Table des matières

1. Contenu

Ce document à pour but d'expliquer et de définir l'usage de la classe **`SimpleListingHelper`** de l'outil **UniversalWeb**. Dans ce document les termes employés de liste et listing font référence à un tableau au sens HTML défini par le tag **`<table>`**.

Pour obtenir la version de cette classe, tapez **`echo SimpleListingHelper::VERSION`**.

2. Définition

La classe **`SimpleListingHelper`** a pour rôle d'aider à la création de listes et listings simple sous forme de tables HTML. Elle est **uniquement** constituée de méthodes statiques qui permettent d'accéder directement à ses fonctionnalités sans instanciation via un objet. Veuillez vous reporter à la documentation de la classe **`UniversalList`** pour découvrir sa mise en place via un exemple concret.

3. Méthodes statiques

createCol

Création d'une colonne de listing.

Description

```
array createCol(array $parametres);
```

Liste des paramètres

`$parametres` : tableau de paramètres contenant la description de la colonne. Les paramètres à renseigner sont :

- **`name`** : (string) libellé à afficher sur l'entête de la colonne.
- **`size`** : (string) taille de la colonne en % (défaut 10%)
- **`align`** : (string) alignement du contenu de la colonne (left* / center / right)
- **`tri`** : (string) champ SQL de la table utilisé pour trier la colonne (vide par défaut).
- **`sens`** : (string) sens du tri de la colonne triée (ASC* / DESC)
- **`title`** : (string) info-bulle sur la colonne qui apparaît au passage de la souris (vide par défaut)
- **`header`** : (booléen) définit la colonne comme entête pour la ligne (**`false*`**)
- **`css`** : (string) code CSS du libellé de l'entête de la colonne (vide par défaut)

*Valeurs par défaut

Valeurs de retour

La méthode renvoie un tableau identique enrichi des éventuels paramètres non renseignés par affectation des valeurs par défaut.

Exemple

Ici le listing **`$cols`** reçoit dans son indice **`titre`** le descriptif de la colonne qui aura pour libellé 'Titre'.

```
$cols['titre'] = SimpleListingHelper::createCol(array
                    ('name' => 'Titre',
                    'size' => 40,
                    'tri' => 'titre',
                    'header' => true));
```

Note
Aucun paramètre n'est obligatoire. Dans le pire des cas, si **parametre** est vide, la colonne ne permettra pas de tri, aura comme le titre 'Nom Colonne ?' (pour bien marquer l'oubli), une taille de 10% et un alignement du texte à gauche (left).

getSize

Renvoie la taille totale du listing en pourcentage par addition de la taille de chaque colonne définie.

Description
```
integer getSize(array $listing);
```

Liste des paramètres
Listing : le listing (collection de colonnes sous forme de tableau).

Valeurs de retour
Largeur totale de la liste en pourcentage (100% étant la largeur totale).

Exemple
```
echo SimpleListingHelper::getSize($cols);
```

sizeAlarm

Envoi un message en DEBUG_ lorsque la taille totale des colonne est différente de 100%.

Description
```
sizeAlarm(array $listing);
```

Liste des paramètres
Listing : le listing (collection de colonnes sous forme de tableau).

Valeurs de retour
Aucun retour directe. La méthode envoi cependant l'information de débordement à l'application qui va l'exploiter via la fonction **DEBUG_PRINT_()**[1].

Exemple
```
SimpleListingHelper::sizeAlarm($cols);
```

getParams

Gère tous les paramètres de gestion du listing (tri, sens du tri, page, etc.). Cette méthode **obligatoire** doit être appelée après la création des colonnes et avant avant l'appel aux données. C'est cette méthode qui fait vivre le listing.

Description
```
getParams(string $idColonne, array &$cols, integer &$page, string &$tri,
string &$sens)
```

Liste des paramètres

[1] Voir le document « Mécanismes »

idColonne : identifiant de la colonne par défaut utilisée pour le premier tri. Le sens du tri de la colonne est pris en compte. Si aucun champ de tri n'est disponible pour la colonne par défaut, l'affichage de la liste peut conduire à une erreur SQL dans le cas où la requête qui sera exécutée utiliserait une clause **ORDER BY**.
cols : notre tableau de colonnes.
page : la page à afficher en cours.
tri : le champ SQL de tri en cours.
sens : le sens du tri en cours (**ASC** ascendant ou **DESC** descendant).

Valeurs de retour
Après gestion, la méthode modifie les paramètres **cols**, **page**, **tri** et **sens** selon l'usage que l'utilisateur a fait de la liste (changement de page, tri sélectionné, etc.).
Si la colonne par défaut n'existe pas la méthode arrête le programme et affiche le message suivant : '*Default column does not exist*'.

Exemple
```
SimpleListingHelper::getParams('name', $cols, $page, $tri, $sens);
```

Note
Cette fonction est le cœur de la gestion du listing. Elle doit être appelée après la création des colonnes et avant l'appel aux données.

drawTotal

Affiche le total de références (données) gérées par le listing.

Description
```
drawTotal(integer $nombreLignes);
```

Liste des paramètres
nombreLignes : nombre d'éléments total gérés par le listing. Par exemple un listing utilisateur peut gérer 1200 utilisateurs et ne montrer qu'une seule page de 30 utilisateurs.

Valeurs de retour
Affichage du nombre de références trouvées.

Exemple
```
SimpleListingHelper::drawTotal($totalLignes);
```

Note
Cette méthode n'a pour vocation que l'affichage de cette information.

drawHead

Dessine l'entête du listing.

Description
```
drawHead(array $cols, string $tri, string $sens [, string $css])
```

Liste des paramètres
cols : le tableau de colonnes.
tri : le champ SQL de tri en cours.
sens : le sens du tri en cours (**ASC** ascendant ou **DESC** descendant).

css : code CSS pour l'habillage de l'entête (optionnel).

Valeurs de retour
Aucun.

Exemple
```
SimpleListingHelper::drawHead($cols, $tri, $sens);
```

drawBody

Dessine le corps du listing

Description
```
drawBody(array $cols, array $listing, integer $page)
```

Liste des paramètres
cols : le tableau de colonnes.
listing : le tableau contenant les données à afficher.
page : la page du listing à afficher.

Valeurs de retour
Aucune.

Exemple
```
echo '<table class="table table-hover">';
  //affichage de l'entete
  SimpleListingHelper::drawHead($cols, $tri, $sens);
  //affichage du corps du tableau : donnees
  SimpleListingHelper::drawBody($cols, $listing, $page);
echo '</table>';
```

Notes

- La couleur du background de la ligne est donnée par le champ **line-color** des données à afficher. Voir exemple de la classe **UniversalList**.
- Pour permettre à une ligne du tableau d'être **draggable** (c'est-à-dire lui autoriser à être déplaçable à la souris par une opération de Drag & Drop) le champ **line-draggable** des données à afficher doit être saisi. Voir exemple de la classe **UniversalList**.
- Pour permettre à une ligne du tableau d'être **droppable** (c'est-à-dire lui autoriser à recevoir un drop suite à une opération de Drag & Drop) le champ **line-droppable** des données à afficher doit être saisi. Voir exemple de la classe **UniversalList**.
- Pour passer un attribut **info** aux lignes <tr> du tableau il faut utiliser le champ **line-info** des données à afficher. Voir exemple de la classe **UniversalList**.

Exemples :

```
foreach($listing as $indice => $ligne) {
    $listing[$indice]['line-color'] = 'text-danger';    //colore la ligne en rouge
    $listing[$indice]['line-droppable'] = true;         //rend la ligne droppable
    $listing[$indice]['line-draggable'] = true;         //rend la ligne draggable
    $listing[$indice]['line-info'] = 12;                //envoie une information au <tr>
}

Affichera
<tr class="text-danger" draggable="true" dropper="true" info="12">
```