listes meetic

décembre 2006 version 1.0.34

Table des matières

1. Présentation

Le site manipule un ensemble de listes qui sont hétérogènes aussi de la façon de les générer, de les conserver, et de les manipuler (naviguer).

Le but est d'arriver à rationaliser les listes, sans perdre de fonctionnalités, ni performances.

2. Principe

Créer une librairie d'outils manipulant les listes, de la génération, conservation, et navigation.

De façon à simplifier, une liste est une table de clés. On a donc besoin de décrire comment alimenter cette table, éventuellement comment la filtrer (définir une vue), comment l'enrichir, comment naviguer.

La description de la liste se trouve dans un fichier d' entête (.inc) et sera incluse dans les scripts manipulant la liste décrite.

3. Alimenter une liste

```
/**
 * creates a new list
 *
 * @param string  uniq Id of list
 * @param array   functions array
 * @param array   optional, arguments array of function
 * @param array   optional, count function
 * @param string  ex: type MLIST_SESSION MLIST_FILE MLIST_DB MLIST_RAM
MLIST_REQUEST MLIST_TEST MLIST_INTERNAL
 * @param integer optional, time to live of the list in sec
 * @param integer optional, minimun size of the list
 * @param integer optional, maximum size of the list
 *
 * functions is used to fill the list, they must return keys array.
 *
 * @return array the new list
 */
function &mlist_new($listId, $func, $args = null, $counter = null, $type = null,
$ttl = 0, $chunkSize = 0, $minSize = 0, $maxSize = 0)

/**
 * refreshes the list
 *
 * @param array the list
 * @param mixed optional, a function or functions array
 * @param array optional, arguments to functions
 * @param bool  optional, force refresh
 *
 * @return bool true if the list has to be refreshed
 */
function mlist_refresh(&$list, $func = null, $args = null, $dirty = false)
```

4. Filtrer la liste

```
/**
 * adds a filter function to list
 * a filter function takes a keys array and returns a keys array
 *
 * @param array the list
 * @param string function name
 * @param string optional, id of filter if differs of function name
 * @param mixed optional, userdata
 *
 */
function mlist_filterAdd(&$list, $func, $filterId = null, $userdata = null)
```

5. Enrichir la liste

```
/**
 * adds an enricher function to list
```

```
 * an enricher function takes a keys array and return associative array key =>
row
 *
 * @param array the list
 * @param string function name
 * @param string optional, id if differs of function name
 * @param mixed optional, user data
 *
 */
function mlist_enricherAdd(&$list, $func, $enricherId = null, $userdata = null)
```

6. Définir une vue

```
/**
 * adds a view to list
 *
 * @param array the list
 * @param string id of view
 * @param mixed optional, id of filter or array of filter id
 */
function mlist_viewAdd(&$list, $viewId, $filterId = null, $ttl = 0,  $type =
MLIST_TYPE_INTERNAL) {
```

7. Définir un navigateur

```
/**
 * adds a navigator to list
 *
 * @param array the list
 * @param string id of navigator
 * @param string id of view
 * @param type of navigation MLIST_NAV_PAGE, MLIST_NAV_SHUFFLE, MLIST_NAV_DEFAULT
 * @param number size
 * @param string optional, wrapper before each column name
 */
function mlist_navigatorAdd(&$list, $navId, $viewId, $enricherId, $navType =
MLIST_NAV_DEFAULT, $navSize = 0, $wrapper = null)
```

8. Naviguer dans la liste

```
/**
 * navigates in the list
 *
 * @param array the list
 * @param string id of navigator
 * @param mixed cursor key
 * @param optional, a map function to finalize the response
 *
 * @return array an associative array whith keys rows, cols and index for type
PAGE
```

```
 */
function mlist_navigate(&$list, $navId, $cursorKey = null, $map = null)
```

La fonction mlist_navigate valorise un tableau contenant le résultat de la navigation. Suivant le type de navigation le résultat peut changer.
On trouve toujours la propriété 'rows', ce tableau a pour éléments un ensembles de lignes valorisées par l'enrichisseur.
De même on trouve la propriété 'cols', ce tableau est une translation de rows, c'est donc un tableau associatif ayant pour clé le nom des colonnes (de l'enrichisseur) et comme valeur un tableau des occurrences de ces colonnes.

Pour les navigations de type PAGE, on a une propriété supplémentaire 'index'. Cette propriété est un tableau contenant tous les informations de pagination. (voir annexe variables de pagination)

9. Conserver une liste

Ces fonctions doivent être définies au moment de l'implémentation. Elles permettent de conserver une mémoire de liste entre 2 navigations. Si elles ne sont pas définis, la liste et ses dépendance seront générées à chaque navigation. Le nom des fonctions dépend du type de la liste. Le type est simplement ajouté au nom de la fonction.

```
/**
 * saves a list
 *
 * @param string list reference
 */
function mlist_save***TYPE***(&$list)

/**
 * saves a view
 *
 * @param string list reference
 * @param string view id
 */
function mlist_viewSave***TYPE***(&$list, $viewId)
```

10. Charger une liste

Ces fonctions doivent être définies au moment de l'implémentation. Elles permettent de conserver une mémoire de liste entre 2 navigations. Si elles ne sont pas définis, la liste et ses dépendance seront générées à chaque navigation. Le nom des fonctions dépend du type de la liste. Le type est simplement ajouté au nom de la fonction.

```
/**
 * loads a list
 *
```

```
 * @param string the list
 */
function mlist_load***TYPE***(&$list)

/**
 * loads a view
 *
 * @param string id of list
 */
function mlist_viewLoad***TYPE***(&$list, $viewId)
```

11.Annexes

1. Prototypage fonction d'alimentation de liste

```
/**
 * returns an associative array with count and keys index
 * index count is the total of elements in the list
 * index keys is array of keys of elements
 *
 * @param mixed ... optional arguments
 * @return array ['count'=>0, 'key'=>array()]
 */
function select()
```

exemple :

```
function select($first = null, $lenght = null) {
	$res = array('a', 'd', 'e', .... );
	if($first != null)
		$res = array_slice($res, $first, $lenght);
	return array('count'=>count($res), 'keys'=>$res);
}
```

Si la liste est découpée en chunk (si on a spécifié un chunkSize) :

```
/**
 * returns an associative array with offset, count and keys index
 * index offset is the position in the list of the chunk
 * index count is the total of elements in the list
 * index keys is array of keys of the chunk
 *
 * @param mixed ... optional arguments
 * @return array ['offset'=>0, 'count'=>0, 'key'=>array()]
 */
function selectChunked($cursor, $lenght, $arg = null)
```

exemple :

```
function selectChunked($cursor, $lenght, $arg = null) {
	$res = array('a', 'd', 'e', 'f', ...);
	$count = count($res);
	if(is_string($cursor)) {
		$cursor = array_search($cursor, $res);
		if($cursor === false)
			$cursor = 0;
	}
	$res = array_slice($res, $cursor, $lenght);
	$size = count($res);
	return array('offset'=>$cursor, 'count'=>$count, 'keys'=>$res);
}
```

2. Prototypage fonction d'enrichissement

```
/**
 * enrichers an array of keys
 *
 * @param array $keys
 * @return array an array with index key of associative array of columns
 */
function &enricher($keys)
```

exemple :

```
function &enricher($keys) {
	$res = array();
	foreach($keys as $key)
		$res[$key] = array('key'=>$key, 'col1'=>$key.'_col1',
'col2'=>$key.'_col2', 'col3'=>$key.'_col3', 'col4'=>$key.'_col4');
	return $res;
}
```

3. Prototypage fonction de filtre

```
/**
 * returns an filtered array ok keys
 *
 * @param array $keys
 * @return array
 */
function filter($keys)
```

exemple :

```
function filter($keys) {
	$res = array();
	foreach($keys as $r)
		if($r >= 'a' && $r <= 'z')
			$res[] = $r;
	return $res;
}
```

4. Prototypage fonction de map

```
/**
 * maps an array of rows
 *
 * @param array $rows
 * @return array returns an array of rows
 */
function map($rows)
```

exemple :

```
function myMap($rows) {
	foreach($rows as $k=>$r) {
		$r['map'] = 'map';
		$rows[$k] = $r;
```

```
	}
	return $rows;
}
```

5. Variables de pagination

La fonction de navigation valorise une propriété 'index' dans son tableau de résultat. Cette index est lui même un tableau associatif contenant :

```
'index_currentnumber'   : numéro de la page courante
'index_currentkey'      : clé du premier élément de la page courante
'index_firstnumber'     : numéro de la première page
'index_firstkey'        : clé du premier élément de la première page
'index_previousnumber'  : numéro de la page précédente
'index_previouskey'     : clé du premier élément de la page précédente
'index_beforenumber'    : tableau de numéros de page précédant la page courante
'index_beforekey'       : tableau de clés du premier élément des pages précédentes
'index_lastnumber'      : numéro de la dernière page
'index_lastkey'         : clé du premier élément de la dernière page
'index_nextnumber'      : numéro de la page suivante
'index_nextkey'         : clé du premier élément de la dernière page
'index_afternumber'     : tableau de numéros de page suivant la page courante
'index_afterkey'        : tableau de clés du premier élément des pages suivantes
```

6. Exemple

Le code php :

```
<?php
include("sdk/list/mlist.inc");
function select($first = null, $lenght = null) {
	$res = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
39, 40, 41, 42, 43, 44, 45, 46, 47);
	if($first != null)
		$res = array_slice($res, $first, $lenght);
	return array('count'=>count($res), 'keys'=>$res);
}

function enricher($keys) {
	$res = array();

	foreach($keys as $key)
		$res[$key] = array('key'=>$key, 'col1'=>$key.'_col1',
'col2'=>$key.'_col2', 'col3'=>$key.'_col3', 'col4'=>$key.'_col4');
	return $res;
}

$list = &mlist_new('test', 'select');
mlist_enricherAdd($list, 'enricher');
mlist_viewAdd($list, 'view1');
mlist_navigatorAdd($list, 'nav1', 'view1', 'enricher1', MLIST_NAV_PAGE, 5);
```

```
$tmpl = yats_define('list.htm');

$cursor = $_GET['cursorKey'];

$res = mlist_navigate($list, 'nav1', $cursor);

yats_assign($tmpl, $res['cols']);
yats_assign($tmpl, $res['index']);

echo yats_getbuf($tmpl);
?>
```

Le code html :

```
html>
<head>
</head>
<body>
<table border="1">
      <tr>
            <td valign="top">
                  <u>vue1: nav1: page</u>
                  total: {{count}}<br/>
                  {{section:nav1}}{{col1}} <a
href="list.php?cursorKey={{list_cursorkey}}">view</a><br>{{/section:nav1}}
            </td>
      </tr>
      <tr>
            <td valign="top">
                  {{section:s_previous autohide="yes"}}<a
href="list.php?cursorKey={{index_previouskey}}">
                  previous</a>{{/section:s_previous}}
                  Pages :
                  {{section:s_indexFirst autohide="yes"}}<a
href="list.php?cursorKey={{index_firstkey}}">
                  {{index_firstnumber}}</a> ...{{/section:s_indexFirst}}

                  {{section:s_indexBefore autohide="yes"}}<a
href="list.php?cursorKey={{index_beforekey}}">
                  {{index_beforenumber}}</a>|{{/section:s_indexBefore}}

                  {{section:s_indexCurrent
autohide="yes"}}{{index_currentnumber}}{{/section:s_indexCurrent}}

                  {{section:s_indexAfter autohide="yes"}}|<a
href="list.php?cursorKey={{index_afterkey}}">
                  {{index_afternumber}}</a>{{/section:s_indexAfter}}

                  {{section:s_indexLast autohide="yes"}} ... sur <a
href="list.php?cursorKey={{index_lastkey}}">
                  {{index_lastnumber}}</a>{{/section:s_indexLast}}

                  {{section:s_next autohide="yes"}}<a
href="list.php?cursorKey={{index_nextkey}}">
```

```
                next</a>{{/section:s_next}}
            </td>
        </tr>
</table>


</html>
```