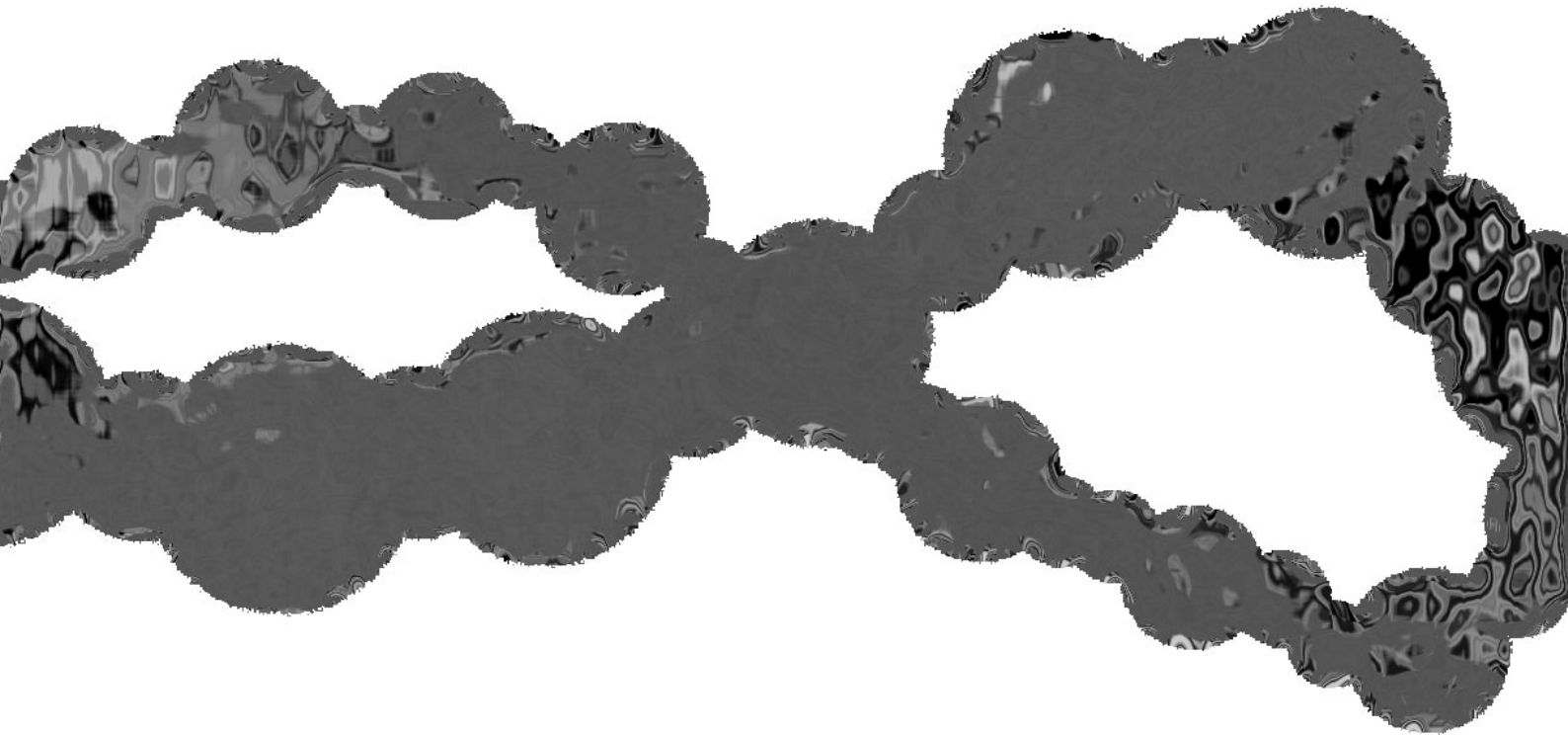


S PHP imple DB



Simple mapping database to objects

Documentation d'utilisation de PHP Simple DB

Introduction et conseil d'utilisation.....	- 3 -
Manipulation simple d'un objet	- 4 -
Manipulation simple d'une collection.....	- 5 -
Documentation détaillée des objets.....	- 6 -
Méthode getData_field(\$sName)	- 7 -
Méthode setData_field(\$sName, \$sValue)	- 8 -
Méthode getData_object(\$sName).....	- 9 -
Méthode setData_object(\$sName, \$oValue).....	- 10 -
Méthode getData_collection(\$sName).....	- 11 -
Méthode load(\$oData = NULL).....	- 12 -
Méthode save (\$oData = NULL, \$bForceCreate = false)	- 13 -
Méthode delete(\$bDeleteCollections = false)	- 14 -
Méthode getData_infos().....	- 15 -
Méthode getData_values()	- 16 -
Méthode getData_key().....	- 17 -
Documentation détaillée des collections	- 18 -
Méthode setLimit(\$iLimit_page, \$iLength).....	- 18 -
Méthode setOrderBy(\$sOrder, \$bAsc = true)	- 19 -
Méthode count(\$oData).....	- 20 -
Méthode load(\$oData = NULL).....	- 21 -
Méthode save().....	- 22 -
Méthode delete(\$bDeleteCollections = false)	- 23 -
Méthode addObject(\$oObject).....	- 24 -
Méthode delObject(\$oObject, \$bDeleteCollections = false)	- 25 -
Méthode getObjects().....	- 26 -
Méthode sortByField(\$sName, \$sBy = «asc», \$sType = SORT_NUMERIC).....	- 27 -
Méthode moveOrderField(\$sName, \$iId, \$sBy = «asc»)	- 28 -
Méthode implodeField(\$sSeparator, \$sField)	- 29 -
Fin.....	- 30 -

Introduction et conseil d'utilisation

Introduction :

PHP SIMPLE DB est un outil d'aide à la manipulation de données en PHP.
Cet outil vous permettra de générer un ensemble de classes représentatives de votre modèle métier.
La solution se présente sous la forme d'un portail web.

Conseil d'utilisation :

Nous vous conseillons d'utiliser de manière globale l'objet d'accès à la base de données.

```
<?php
include_once( 'phpsimpledb.class.php' );
Global $oBDD ;
$oBDD = new PHPSIMPLEDB();
?>
```

Les données de connexion (serveur, base, user, mot de passe) se trouvent dans la classe «phpsimpledb.class.php».

Si vous modifiez directement les données de connexion dans le fichier, celles-ci pourront être perdues lors du prochain re-déploiement.

Si vous souhaitez utiliser des constantes à la place de valeur en claire, modifier votre fichier source (une fois généré) et cocher la case (ne pas générer la classe «phpsimpledb.class.php») lors de votre prochaine génération.

Manipulation simple d'un objet

Dans l'exemple suivant nous montrons comment mettre à jour une adresse email d'un utilisateur en fonction de son numéro d'identifiant.

```
<?php

//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//création et chargement du "user"
$oUser = $oBDD->getObject( "users" );
$oUser->load( 645 );

//mise à jour du 'user'
$oUser->setData_field( "email", "admin@phpsimpledb.com" );
$oUser->save();

?>
```

Manipulation simple d'une collection

Dans l'exemple suivant nous montrons comment mettre à jour une collection d'utilisateurs en fonction de leurs niveaux d'habilitations.

```
<?php

//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//création et chargement de la collection de "users"
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->load( array( "level > 80" ) );

//mise à jour des "users" de la collection
foreach( $oUsers_col->getObjects() as $oUser ){
    $oUser->setData_field( "level", 1 );
    $oUser->save();
}

?>
```

Documentation détaillée des objets

Prés requis :

Afin de pouvoir utiliser correctement les exemples suivants, nous vous recommandons de veiller au bon établissement des prés requis ci-dessous.

Note : Il n'est pas nécessaire de mettre en place ces prés requis.

Vous devez avoir créé une base de données avec les tables suivantes :

Users :

- Id : bigint, auto incrémentation
- Name : chaîne(255)
- Level : entier

Produits :

- Id : bigint, auto incrémentation
- Name : chaîne (255)
- Position : int,
- Users_id : bigint, clé étrangère pointant sur la table «users»

Script SQL :

```
CREATE TABLE `test`.`users` (  
  `id` BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY ,  
  `name` VARCHAR( 255 ) NOT NULL ,  
  `level` INT NOT NULL  
) ;  
  
CREATE TABLE `test`.`produits` (  
  `id` BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY ,  
  `name` VARCHAR( 255 ) NOT NULL ,  
  `position` INT NULL ,  
  `users_id` BIGINT NOT NULL  
) ;
```

Utiliser l'application web «PHPSIMPLEDB» pour créer le schéma de base pour toutes les tables, puis lier la table «produits» à la table «users» par l'intermédiaire du champ «users_id».

Générer l'archive et importer la dans votre projet.

Si vous avez des difficultés à réaliser cette archive, suivez le tutorial «utilisation de l'application web».

```
//instanciation globale  
include_once( 'phpsimpledb.class.php' );  
Global $oBDD;  
$oBDD = new PHPSIMPLEDB();
```

Méthode `getData_field($sName)`

Description :

Permet la récupération d'un attribut d'un objet.

Paramètre :

`$sName`, chaîne : Renseigne le nom de la propriété de l'objet

Valeur de retour :

Retourne une valeur contenant la propriété demandée,

Retourne «NULL» si la propriété est inexistante ou si elle n'a pas de valeur

L'exemple suivant affiche le nom d'un «user».

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//chargement du "user"
$oUser = $oBDD->getObject( "users" );
$oUser->load( 21 );

//affichage du nom
print( $oUser->getData_field( "name" ) );
?>
```

Méthode setData_field(\$sName, \$sValue)

Description :

Cette méthode ajoute ou met à jour une propriété d'un objet

Paramètres :

\$sName, chaîne : Représente le nom du champ à mettre à jour

\$sValue, chaîne ou numérique : représente la valeur à mettre à jour

Valeur de retour :

Retourne «true» si le champ a été mis à jour

Retourne «false» si le champ n'a pas été mis à jour

L'exemple suivant met à jour le nom d'un «user» :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//chargement du "user"
$oUser = $oBDD->getObject( "users" );
$oUser->load( 21 ) ;

//mise à jour du nom
$oUser->setData_field( "name", "durant" );

//enregistrement du "user"
$oUser->save();
?>
```

Méthode `getData_object($sName)`

Description :

Cette méthode retourne un objet qui se trouve implicitement rattaché à un parent. Pour exemple, dans un site de petites annonces immobilières, une maison (produit) sera rattachée à une personne (user).

Paramètre :

`$sName`, chaîne : Indique le nom du champs qui identifiera l'objet rattaché

Valeur de retour :

Retourne un objet si celui-ci existe dans l'objet parent

Retourne «NULL» si l'objet n'a pas été trouvé

L'exemple suivant retour le «user» à un produit

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupère un produit
$oProduit = $oBDD->getObject( "produits" );
$oProduit->load( 12 );

//récupère l'objet « user » rattaché au produit
$oUser = $oProduit->getData_object( "users_id" );

?>
```

Méthode setData_object(\$sName, \$oValue)

Description :

Cette méthode ajout un objet «a» à un autre objet «b» qui contient implicitement une collection de type «b». Pour exemple, dans un site de petites annonces immobilières, une maison (produit) sera rattachée à une personne (user).

Paramètres :

\$sName, chaîne : Indique le nom du champs qui identifiera l'objet rattaché
\$oValue, objet : Indique l'objet à rattacher

Valeur de retour :

Retourne «true» si l'ajout a été effectué avec succès
Retourne «false» si l'objet n'a pu être ajouté

L'exemple suivant ajout un produit à un «user» :

```
<?php

//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupération du "user"
$oUser = $oBDD->getObject( "users" );
$oUser->load( 21 );

//création du produit
$oProduit = $oBDD->getObject( "produits" );
$oProduit->setData_field( "name", "jolie maison" );

//ajout du produit au "user"
$oProduit->setData_object( "users_id", $oUser );

//sauvegarde du produit
$oProduit->save();

?>
```

Note : Si vous tentez de sauvegarder un nouvel objet en base, assurez-vous que toutes les valeurs obligatoires sont renseignées. Si vous oubliez de renseigner les champs obligatoires, votre tentative de sauvegarde échouera.

Méthode `getData_collection($sName)`

Description :

Cette méthode retourne une collection appartenant à un objet.

Paramètre :

`$sName`, chaîne : Indique le nom du champ qui identifiera la collection

Valeur de retour :

Retourne une collection d'objet si celle-ci est présente dans l'objet (elle peut être vide, si aucun enfant n'est présent).

Retourne «NULL» si la collection n'a pas été trouvée.

L'exemple suivant affiche le nom des produits d'un «user» :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupération du "user"
$oUser = $oBDD->getObject( "users" );
$oUser->load( 21 );

//récupération de la collection des produits
$oCol_produits = $oUser->getData_collection( "produits" );

//affichage des noms des produits
foreach( $oCol_produits->getObjects() as $oProd ){
    print( "nom du produit : ".$oProd->getData_field( "name" )."<br/>" );
}
?>
```

Méthode load(\$oData = NULL)

Description :

Cette méthode alimente les propriétés d'un objet à partir de la base de données.

Paramètre :

\$oData (facultatif), (chaîne ou numérique, table de clé / valeur) : représente soit la clé primaire de l'objet (chaîne ou numérique), soit un tableau de clé / valeur représentant un filtre de recherche ou la première occurrence sera renvoyée.

Valeur de retour :

Retourne «true» si le chargement c'est bien passé.

Retourne «false» si le filtre de recherche «\$oData» n'a fait remonter aucun résultat.

Note : Dans certain cas, où la clé primaire d'un objet n'est pas un entier, certaines instabilités pourront parvenir. Nous vous recommandons de préférer la mise en place d'une clé primaire numérique pour chacune de vos tables.

L'exemple suivant recherche et affiche le nom d'un user :

```
<?php
//instanciation globale
include_once( 'phpsimplifiedb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupération du "user"
$oUser = $oBDD->getObject( "users" );
$oUser->load( 12 ) ;

//affichage du nom du "user"
print( $oUser->getData_field( "name" ) );

?>
```

Méthode save (\$aData = NULL, \$bForceCreate = false)

Description :

Cette méthode sauvegarde les propriétés d'un objet dans la base de données.

Paramètres :

`$aData` (facultatif), tableau clé / valeur : Ce paramètre peut contenir des paires clé / valeur ou clé / objet. Les propriétés de l'objet seront mises à jour avant son enregistrement.

`$bForceCreate` (facultatif), booléen : Ce paramètre est utilisé lors de la création d'objet ne comportant pas de clé primaire numérique. En mettant le paramètre à «true», l'objet sera créé dans la base de données.

Valeur de retour :

Aucun paramètre de retour.

L'exemple suivant illustre l'enregistrement d'un produit avec une mise à jour de certaines propriétés :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupération du produit
$oProduit = $oBDD->getObject( "produits" );
$oProduit->load( 2 );

//création d'un nouveau "user"
$oUser = $oBDD->getObject( "users" );
$oUser->setData_field( "name", "John Do" );
$oUser->setData_field( "level", 80 );

//mise à jour du produit
$oProduit->save( array( "name"=>"Jolie maison de campagne", "users_id"=>$oUser ) );
?>
```

Méthode delete(\$bDeleteCollections = false)

Description :

Cette méthode supprime un objet de la base de données.

Paramètre :

\$bDeleteCollection (facultatif), booléen : Ce paramètre permet le déclenchement de la suppression des objets en cascade, toutes les collections des objets enfants seront elles aussi supprimées (attention, la notion de niveau de suppression n'existe pas).

Valeur de retour :

Retourne «true» si l'objet a pu être supprimé.

Retourne «false» si l'objet n'a pu être supprimé (si celui-ci n'est pas présent dans la base de donnée, «false» sera renvoyé).

L'exemple suivant illustre la suppression d'un produit :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupération du produit
$oProduit = $oBDD->getObject( "produits" ) ;
$oProduit->load( 12 ) ;

//supprime le produit
$oProduit->delete() ;

?>
```

Méthode `getData_infos()`

Description :

Cette méthode retourne un tableau clé / valeur contenant les informations liées à la table. Ce tableau contient le nom de la table, le nom de la clé primaire et la description de la table.

Valeur de retour :

Retourne un tableau associatif clé / valeur contenant les informations de la table.

L'exemple suivant affiche le nom et la clé primaire de la table des produits :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupération d'un objet "produit"
$oProduit = $oBDD->getObject( "produits" );

//récupération des informations
$oData_table = $oProduit->getData_infos();

//affichage des informations
print( "Table : ".$oData_table[ "table" ]."<br/>" );
print( "Clé : ".$oData_table[ "primarykey" ]."<br/>" );
?>
```

Méthode `getData_values()`

Description :

Cette méthode retourne un tableau clé / valeur contenant toutes les valeurs de l'objet indexé selon le nom du champ.

Valeur de retour :

Retourne un tableau associatif clé / valeur contenant le nom des champs en index et leurs valeurs associées.

L'exemple suivant affiche l'ensemble des propriétés d'un objet produit :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupération du produit
$oProduit = $oBDD->getObject( "produits" );
$oProduit->load( 13 );

//récupération de toutes les valeurs du produit
$oValues = $oProduit->getData_values();

//affichage des valeurs
foreach( $oValues as $sKey=>$sValue ){
    print( $sKey." : ".$sValue."<br/>" );
}
?>
```

Méthode `getData_key()`

Description :

Cette méthode renvoie la clé primaire de l'objet. Si celle-ci est inexistante, «NULL» sera renvoyé.

Valeur de retour :

Retourne la valeur de la clé primaire de l'objet.

Retourne «NULL» si la valeur de la clé primaire de l'objet est nulle.

L'exemple suivant affiche la clé primaire d'un produit :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupération du produit
$oProduit = $oBDD->getObject( "produits" );
$oProduit->load( 13 );

//affiche la clé primaire
print( "Clé : ".$oProduit->getData_key() );
?>
```

Documentation détaillée des collections

Méthode setLimit(\$iLimit_page, \$iLength)

Description :

Permet la limitation du nombre d'objets retournés par la collection.

Paramètres :

`$iLimit_page`, entier : Ce paramètre attend un numéro de page (minimum 1)

`$iLength`, entier : Ce paramètre détermine le numéro de page à renvoyer

Valeur de retour :

Aucun paramètre de retour.

L'exemple suivant récupère les 10 premiers «users» en fonction de leurs niveaux d'autorisation :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//chargement des 10 premiers "users" en fonction de "level = 1"
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->setLimit( 1, 10 );
$oUsers_col->load( array( "level = 1" ) );
?>
```

Méthode `setOrderBy($sOrder, $bAsc = true)`

Description :

Cette méthode paramètre un ordre de classement des lignes d'une collection. Le classement peut se faire selon un champ, dans l'ordre ascendant ou descendant.

Paramètres :

`$sOrder`, chaîne : Détermine le champ dans lequel les objets de la collection doivent être classés.

`$bAsc` (facultatif), booléen : Détermine le sens de tri de la collection, «true» pour ascendant. Par défaut l'ordre de classement sera ascendant si le deuxième paramètre est omis.

Valeur de retour :

Aucun paramètre de retour.

L'exemple suivant récupère une liste de «users» classés par ordre alphabétique sur le champ «name» :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//chargement des "users" par ordre alphabétique
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->setOrderBy( "name" );
$oUsers_col->load();

?>
```

Méthode count(\$oData)

Description :

Permet de compter le nombre d'élément d'une expression.

Paramètre :

\$oData, (chaîne ou tableau de chaîne) : Le paramètre sera soit de type tableau de valeur, soit de type chaîne. Si celui-ci est de type tableau, les éléments du tableau seront assemblés avec l'opérateur SQL «AND».

Valeur de retour :

Retourne un entier représentant le résultat du comptage des éléments.

L'exemple suivant compte le nombre de «users» ayant un niveau d'autorisation égale à 1 :

```
<?php
//instanciation globale
include_once( 'phpsimplifiedb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//compte le nombre de "users" non valides
$oUsers_col = $oBDD->getCollection( "users" );
$iNbr_users = $oUsers_col->count( "level = 1" );

?>
```

Méthode load(\$oData = NULL)

Description :

Permet le chargement d'une collection.

Paramètre :

\$oData (facultatif), (chaîne ou tableau de chaîne) : Le paramètre sera soit de type tableau de valeur, soit de type chaîne. Si celui-ci est de type tableau, les éléments du tableau seront assemblés avec l'opérateur SQL « AND ». Si ce paramètre est omis, tous les enregistrements de la collection seront renvoyés.

Valeur de retour :

Retourne «true» si le chargement de la collection s'effectue correctement.

Retourne «false» si la collection n'a remonté aucun enregistrement.

L'exemple suivant récupère les «users» appartenant au niveau 1 et ne portant pas le nom de « durant » :

```
<?php

//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//chargement des "users" avec "name != 'durant'" et "level = 1"
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->load( array( "name != 'durant'", "level = 1" ) );

?>
```

Méthode save()

Description :

Permet l'enregistrement des objets d'une collection.

Valeur de retour :

Aucun paramètre de retour.

L'exemple suivant modifie le niveau d'autorisation des «users» appartenant au niveau 1 :

```
<?php

//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupère tous les "users" avec "level = 1"
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->load( "level = 1" );

//modification de l'activation
Foreach( $oUsers_col->getObjects() as $oUser ){
    $oUser->setData_field( "level", 2 );
}

//enregistrement des modifications
$oUsers_col->save();

?>
```

Méthode delete(\$bDeleteCollections = false)

Description :

Permet la suppression des objets d'une collection.

Paramètre :

`$bDeleteCollection` (facultatif), booléen : Ce paramètre permet le déclenchement de la suppression des objets en cascade, toutes les collections des objets enfants seront elles aussi supprimées (attention, la notion de niveau de suppression n'existe pas).

Valeur de retour :

Aucun paramètre de retour.

L'exemple suivant supprime la liste des «users» ayant un niveau d'autorisation égale à 0 :

```
<?php
//instanciation globale
include_once( 'phpsimplifiedb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//recupere tous les "users" avec "level = 0"
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->load( "level = 0" );

//supprime les "users"
$oUsers_col->delete();

?>
```

Méthode addObject(\$oObject)

Description :

Permet l'ajout d'un objet à une collection.

Si la collection appartient à un objet (comme «produits» appartient à «users»), l'objet ajouté sera automatiquement connecté à la collection.

Paramètre :

\$oObject, objet : Ce paramètre représente l'objet à ajouter

Valeur de retour :

Retourne «true» si l'ajout à la collection s'est bien passé

Retourne «false» si l'objet passé en paramètre ne correspond pas au type de la collection, ou si l'objet ne contient pas d'identifiant.

L'exemple suivant ajout un produit à la collection des produits d'un «users» :

```
<?php

//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupère tous les "produits" d'un "user"
$oUser = $oBDD->getObject( "users" );
$oUser->load( 2 ) ;

//récupère la collection des produits du "user"
$oProduits_col = $oUser->getData_collection( "produits" );

//création d'un nouveau produit
$oProduit = $oBDD->getObject( "produits" );
$oProduit->setData_field( "name", "Pavillon" );

//ajout du produit à la collection des produits du "user"
$oProduits_col->addObject( $oProduit ) ;
$oProduits_col->save();

?>
```

Méthode delObject(\$oObject, \$bDeleteCollections = false)

Description :

Permet de supprimer un objet d'une collection.

Paramètres :

\$oObject, objet : Ce paramètre représente l'objet à supprimer.

\$bDeleteCollection (facultatif), booléen : Ce paramètre permet le déclenchement de la suppression des objets en cascade, toutes les collections des objets enfants seront elles aussi supprimées (attention, la notion de niveau de suppression n'existe pas).

Valeur de retour :

Aucun paramètre de retour.

L'exemple suivant supprime un utilisateur d'une collection de «users» :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupère une collection de "users"
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->load( "level = 2" );

//récupère le premier enregistrement
$oUsers = $oUsers_col->getObjects();
$oUser = $oUsers[ 0 ];

//supprime le "user"
$oUsers_col->delObject( $oUser );

?>
```

Méthode `getObjects()`

Description :

Cette méthode retourne un tableau contenant une liste d'objets.

Valeur de retour :

Retourne un tableau contenant une liste d'objets.

Si aucun élément n'est trouvé, le tableau renvoyé sera vide.

L'exemple suivant récupère une liste de «users» :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupère une collection de "users"
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->load();

//récupère la liste des "users"
$oUsers = $oUsers_col->getObjects();

?>
```

Méthode `sortByField($sName, $sBy = «asc», $sType = SORT_NUMERIC)`

Description :

Permet de trier une collection sur un champ.

Paramètres :

`$sName`, chaîne : Indique le nom du champ sur lequel sera fait le tri

`$sBy` (facultatif), chaîne : Indique le sens du tri,

«asc», indiquant un tri ascendant

«desc», indiquant un tri descendant

Par défaut ce paramètre est égal à «asc»

`$sType` (facultatif), constante : Renseigne le type de type à effectuer

`SORT_NUMERIC`, indiquant un type de champ numérique

`SORT_STRING`, indiquant un type de champ chaîne

Valeur de retour :

Aucun paramètre de retour.

L'exemple suivant affiche une liste de «users» triée par nom de manière descendante :

```
<?php

//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupère une collection de "users"
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->load();

//Tri descendant sur le champ "name"
$oUsers_col->sortByField( "name", "desc", SORT_STRING );

//affichage des "users"
foreach($oUsers_col->getObjects() as $oUser ){
    print( "nom : ".$oUser->getData_field( "name" )."<br/>" );
}

?>
```

Méthode `moveOrderField($sName, $sId, $sBy = «asc»)`

Description :

Permet de monter ou descendre une valeur d'un champ de type entier dans un objet en respectant l'ordre de la liste.

Paramètres :

`$ sName`, chaîne : Indique le nom du champ qui sera utilisé pour le déplacement

`$sId`, entier : Indique l'identifiant de l'objet

`$sBy` (facultatif), chaîne : Indique le sens du déplacement

«asc», indiquant un tri ascendant

«desc», indiquant un tri descendant

Par défaut ce paramètre est égal à «asc»

Valeur de retour :

Retourne «true» si le déplacement c'est bien passé.

Retourne «false» si la collection ne contient pas d'objet ou si l'objet n'a pas été trouvé.

L'exemple suivant change la position vers le bas d'un produit d'une collection de produit d'un «user» :

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//chargement du "user"
$oUser = $oBDD->getObject( "users" );
$oUser->load( 12 );

//chargement des produits du "users" pour construction de la collection
$oProduits_col = $oUser->getData_collection( "produits" );

//déplacement de la position du produit
$oProduits_col->moveOrderField( "position", 4, "desc" );

//affiche les "produits" avec leurs position
foreach( $oProduits_col->getObjects() as $oProduit ){
    print( "id : ".$oProduit->getData_field( "id" ).", nom : ".$oProduit->
    >getData_field( "name" ).", position : ".$oProduit->
    >getData_field( "position" )."<br/>" );
}

//memorise la collection
$oProduits_col->save();

?>
```

Méthode implodeField(\$sSeparator, \$sField)

Description :

Retourne une chaîne contenant une concaténation du champ désiré.

Paramètres :

\$sSeparator, chaîne : Indique le séparateur de concaténation

\$sField, chaîne : Indique le nom du champ à récupérer

Valeur de retour :

Retourne une chaîne contenant une concaténation des valeurs d'un champ.

L'exemple suivant récupère affiche les noms des «users» en fonction de leurs niveaux d'autorisation, avec une séparation avec des points virgules «;»:

```
<?php
//instanciation globale
include_once( 'phpsimpledb.class.php' );
Global $oBDD;
$oBDD = new PHPSIMPLEDB();

//récupère un collection de "users" non validé
$oUsers_col = $oBDD->getCollection( "users" );
$oUsers_col->load( "level = 2" );

//affichage des nom séparés par des points virgules ";"
print( $oUsers_col->implodeField( ";", "name" ) );
?>
```

Fin

Si cette documentation vous semble incomplète ou si vous avez de mal à comprendre certains points, posez vos questions à : [**contact@phpsimpledb.com**](mailto:contact@phpsimpledb.com)