

Enregistrer et lire des données de fichiers .ini

Par Kyle Katarn



www.openclassrooms.com

*Licence Creative Commons 6 2.0
Dernière mise à jour le 3/05/2011*

Sommaire

Sommaire	2
Partager	1
Enregistrer et lire des données de fichiers .ini	3
Pas de parse_ini_file dans ce tuto	3
Partie 1 : Les bases	4
Découverte du fichier ini	4
Structure d'un fichier .ini	4
Explications	4
Des exemples concrets d'utilisation	5
Lire un fichier ini	5
Récupérer une valeur	6
Création du code PHP	7
Le code PHP	9
Formulaire de sélection	9
Les outils pour l'écriture	11
Le chmod	11
Les fonctions de manipulation de fichiers	11
fopen() et fclose()	11
fwrite()	12
file_put_contents()	12
Effacer pour mieux réécrire	13
Partie 2 : Une page de gestion des fichiers .ini	16
Transformer un array	16
Changer une valeur	16
Ajouter un item ou un groupe	16
Supprimer un item ou un groupe	16
Du fichier à l'array	16
De l'array au fichier	17
Un objet pour tout rassembler	18
Qu'est-ce qu'un objet ?	18
Créer un objet	18
Transmettre dans une SESSION	20
Pour apprendre à se servir des objets	20
Notre objet INI	21
Les méthodes une à une	21
m_fichier()	21
m_groupe()	22
m_item()	22
m_put()	22
m_count()	23
array_groupe()	23
save()	23
clear()	24
s_fichier()	25
s_groupe()	25
s_item()	25
print curseur()	26
print_dossier()	26
print_fichier()	26
m_valeur()	27
Le code PHP	27
Trier les méthodes grâce à l'héritage	31
Le formulaire	31
Le HTML	32
Traitement PHP	34
Conclusion	35
Exemples	36
Enregistrer des notes	36
Système de commentaires	38
Autres idées	39



Enregistrer et lire des données de fichiers .ini

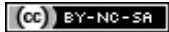
Par



Kyle Katarn

Mise à jour : 03/05/2011

Difficulté : Facile



Ce big-tuto est destiné à tous ceux qui se disent régulièrement : "c'est un peu con de faire une table dans ma base de données juste pour ça", à tous ceux qui n'ont pas de base de données mais qui souhaiteraient enregistrer des informations ; enfin, plus généralement, la sauvegarde INI a des avantages sur la base de données.



Attention : un niveau correct en PHP et en HTML est nécessaire pour pouvoir comprendre ce tutoriel.

Pas de `parse_ini_file` dans ce tuto

Après de vifs commentaires, je me sens obligé de parler de cette fonction en introduction pour en être ensuite débarrassée 😊
Le langage `ini` est flexible. Il n'y a pas de règle stricte pour lire un fichier `ini`. `parse_ini_file` est une fonction qui lit sommairement les fichiers `ini` et renvoie un array simple. Dans ce tuto, nous utiliserons les fichiers `ini` pour le stockage de données, nous allons donc créer une fonction renvoyant un array à deux niveaux en utilisant le plein potentiel de ces fichiers.
Je n'ai pas inventé cette technique mais sachez qu'après lecture du tuto, vous pourrez créer votre propre langage et l'adapter à vos besoins comme bon vous semble.

Partie 1 : Les bases

Cette partie vous apprendra ce qu'est un fichier .ini et comment s'en servir de sauvegarde.

Découverte du fichier ini



Voilà une question qui doit quand même trotter dans quelques têtes :
C'est quoi ces fameux fichiers ini ? 🤔

Alors, pour la petite histoire, ".ini" est leur petit nom (leur extension), ils s'appellent en fait "fichiers d'initialisation" et sont omniprésents dans le système windows ; bien que les plus importants aient été remplacés par des bases de registre dans les dernières versions du système d'exploitation.

Vous allez maintenant découvrir une de ces étranges créatures.

Structure d'un fichier .ini

Comme dit plus haut, les fichiers .ini portent l'extension .ini, mais leur contenu n'est constitué que de texte (un peu comme les fichiers .html ou .php).

Voici à quoi ressemble un fichier .ini :

Citation : exemple.ini

```
[informations générales]
date=12/02/06
candidat=Jean Dupont
note=12
;tout à fait correct
[correction]
date=15/02/06
correcteur=Janine Dujardin
```

Voilà : dans cet exemple, j'ai imaginé qu'on veuille sauvegarder les informations relatives à chaque copie d'un examen dans un fichier. Ceci serait donc le fichier pour la copie de Jean Dupont.

Explications

La première ligne est le nom du groupe : "informations générales", toutes les données qui suivent appartiennent à ce groupe jusqu'à ce qu'on en change.

Les trois lignes qui suivent sont des items, c'est-à-dire des valeurs stockées. Dans date=12/02/06, date est le nom de la variable et 12/02/06, son contenu.

Enfin, la ligne écrite en vert est un commentaire, toute ligne commençant par un point virgule sera ignorée.

Equivalent en php

Code : PHP

```
<?php
$exemple["informations_generales"]=array(
    "date"=>"12/02/06",
    "candidat"=>"Jean Dupont",
    "note"=>"12"
);
//tout à fait correct
$exemple["correction"]=array(
    "date"=>"12/02/06",
    "correcteur"=>"Janine Dujardin"
);
?>
```

Comme je vous le montre ci-dessus, la structure d'un fichier `.ini` est comparable à celle d'un array associatif à deux niveaux.

Des exemples concrets d'utilisation

Personnellement, les fichiers `.ini` m'ont servi dans une galerie d'images pour toutes les options et enregistrements. Ils m'ont aussi été utiles dans la création d'un jeu en ligne pour sauvegarder des informations telles que les compétences d'une race, ou encore les actions possibles sur une case de la map. Mais le `.ini` peut servir à plein d'autres choses :

- livre d'or ;
- chat ;
- système de vote (noter des vidéos par exemple).

Lire un fichier ini

Un fichier `.ini` se lit en pointant un curseur (virtuel) sur une variable. On va dire en langage PHP : "je choisis le fichier `mon_fichier.ini`, je vais dans le groupe `mon_groupe` et dans l'item `mon_item`". On peut maintenant récupérer la valeur à cet endroit et la modifier.



Et en PHP, ça donne quoi??

Effectivement on n'a toujours pas vu l'ombre d'un code PHP. Et ce qui va nous faciliter la vie pour trouver la manière de lire ces fichiers `.ini`, c'est que chaque ligne du fichier correspond à une instruction. Et si je demande quelle est la fonction PHP qui lit un fichier ligne par ligne, vous me répondez tous en coeur ... non ? ... Eh bien c'est `file()`, cette fonction magique qui retourne un tableau où chaque ligne est contenue dans une entrée du tableau. Et le deuxième outil qui va nous être utile : `foreach`. Voici donc comment lire ligne par ligne un fichier :

Code : PHP

```
<?php
$tableau_de_lecture=file($fichier);
foreach($tableau_de_lecture as $ligne)
{
    // Ici on traite une ligne, par exemple, on peut compter la
    longueur de chaque ligne en faisant :
    echo strlen($ligne);
}
?>
```

Et maintenant, il nous faut détecter, pour chaque ligne, s'il y a ouverture de groupe ou un item. Voici un fichier que je vous conseille de prendre pour essayer de trouver le code : http://kyle.katarnls.free.fr/fichier_ini.zip. C'est une liste de membres avec différentes informations pour chacun, en imaginant qu'ils puissent remplir un formulaire sans renseigner tous les champs. Si on avait utilisé une base de données, il y aurait plein de champs vides et il faudrait changer la structure de la table à chaque fois qu'on rajoute une donnée au formulaire. Vous pouvez donc déjà constater un avantage de la *méthode .ini*.

Bon : au boulot, je vous laisse chercher le code pour détecter, je le rappelle, les groupes et les items ...

Secret (cliquez pour afficher)

Code : PHP

```
<?php
if(file_exists($fichier) && $fichier_lecture=file($fichier))
    foreach($fichier_lecture as $ligne)
    {
        if(preg_match("#^\[.+\]#", $ligne))
            //Des crochets au début et à la fin, ça, c'est un
            groupe, pardi !
        else
            if(!preg_match("#^\;#", $ligne)) //S'il n'y a pas de
            point-virgule, ce n'est pas un commentaire
            //Donc c'est un item
        }
    }
else
```

```
die("Le fichier est introuvable ou incompatible");
?>
```

Peut-être que certains n'aiment pas cette façon de coder et préfèrent quand il y a des accolades. Comme j'ai un avis partagé sur le sujet, je vous donne la version aérée :

Secret ([cliquez pour afficher](#))

Code : PHP

```
<?php
if(file_exists($fichier) && $fichier_lecture=file($fichier))
{
    foreach($fichier_lecture as $ligne)
    {
        if(preg_match("#^\[.+\]$#", $ligne))
        {
            //Des crochets au début et à la fin, ça, c'est un
            groupe, pardi !
        }
        else
        {
            if(!preg_match("#^;#", $ligne)) //S'il n'y a pas de
            point-virgule, ce n'est pas un commentaire
            {
                //Donc c'est un item
            }
        }
    }
}
else
{
    die("Le fichier est introuvable ou incompatible");
}
?>
```

Vous savez maintenant différencier les items des groupes à la lecture, nous allons dès le prochain chapitre récupérer notre première valeur.

Récupérer une valeur

Nous allons désormais afficher, à l'aide d'un code PHP, une valeur prélevée dans un fichier `.ini` grâce à vos nouvelles connaissances !

Création du code PHP

Très bien : je vous suggère pour vos essais d'utiliser le fichier dans cette archive :

http://kyle.katarnls.free.fr/fichier_ini.zip, de mettre `membres.ini` au même endroit que votre fichier `.php`.

Avant de vous donner le code de votre fichier `.php` (qui, je le rappelle, doit afficher une valeur du fichier), nous allons réfléchir à ce qu'il y aura dans ce code. Tout d'abord, une valeur est à un endroit précis, on peut dire à une adresse où le nom du fichier serait la ville, le nom du groupe serait la rue et le nom de l'item serait le numéro de la maison. Donc, les trois premières lignes auront pour but de donner l'adresse avec ces trois variables (vous pouvez vous amuser à créer un formulaire pour les entrer). Puis la quatrième ligne initialisera la valeur à `false` : comme ça, si à la fin cette variable vaut toujours `false`, c'est qu'aucune valeur n'a été trouvée, et on pourra afficher un message d'erreur.

Donner l'adresse

On entre le nom du fichier, du groupe, de l'item et on initialise `$valeur` à `false`.

Si après recherche, `$valeur` vaut toujours `false`, cela voudra dire que notre adresse ne mène à aucune valeur.

Code : PHP

```
<?php
$fichier="membres.ini";
$groupe_recherche="Louise";
$item_recherche="code postal";
$valeur=false;
?>
```

Vérifier que le fichier existe

Code : PHP

```
<?php
// Si le fichier existe et qu'on arrive à le lire avec file()
if(file_exists($fichier) && $fichier_lecture=file($fichier))
    // Ici on lance la recherche
else
    echo "Le fichier est introuvable ou incompatible<br />";
?>
```

La recherche

Pour analyser une syntaxe, le mieux, c'est encore les REGEXP. Rien de très compliqué cependant :

Un groupe commence par `[` et termine pas `]`. Donc on utilisera les morceaux suivants :

- `^` signifie commence par un crochet
- `(.+)` c'est le nom du groupe qu'on va récupérer
- `\$` termine par un crochet

Pour l'item, on va juste regarder si la ligne commence par le nom de l'item qu'on cherche donc si la position de `"nom_item="` dans la ligne est égale à 0. La fonction permettant de savoir ça est `strpos` et donc le code est

`strpos($ligne,$item_recherche."")===0`, il y a bien 3 `"` et c'est important parce qu'avec seulement 2. Le triple-égal vérifie que les types soient les même contrairement au double. Un petit tableau devrait aider :

Ceci	renvoie ça
<code>strpos('a=c','a')</code>	0

strpos('b=c','a')	false
0==false	true
0==false	false

Dernière chose, on va utiliser trim(), c'est une fonction de nettoyage, elle enlève tous les espaces au début et à la fin d'une chaîne. Je l'utilise pour détecter les groupes et les commentaires mais pas les items et valeurs car il se pourrait qu'un item commence ou qu'une valeur termine par un espace qu'il faudrait alors conserver.

Attention, voici le plus gros morceau à mettre à la place du commentaire # // Ici on lance la recherche # dans le précédent code :

Code : PHP

```
<?php
foreach($fichier_lecture as $ligne)
{
    $ligne_propre=trim($ligne);
    if(preg_match("#^\[(.+)\]$#", $ligne_propre, $matches))
        $groupe=$matches[1];
    else
        if($ligne_propre[0]!=';')
            if($groupe==$groupe_recherche)
                if(strpos($ligne,$item_recherche."")==0) // Si la
ligne commence par le nom de l'item suivi de =
                    $valeur=end(explode("=", $ligne, 2));
                elseif($ligne==$item_recherche) // Si la ligne
contient juste le nom de l'item
                    $valeur='';
            }
        }
    }
?>
```



On me souffle dans l'oreillette que je n'ai pas défini les fonctions explode() et end()
Merci à Lo-X !



explode("=", \$ligne, 2) découpe la chaîne \$ligne à chaque fois qu'il trouve "=" et renvoie un *array* avec un morceau par entrée. 2 est le nombre maximal de morceau qu'il doit renvoyer. Enfin la fonction end() qui encapsule le tout renvoie la dernière valeur de l'*array* : Dans le cas présent il s'agit de récupérer tout ce qui suit le premier signe =.

Toutes les lignes suivantes font la même chose, à vous de prendre ce que vous comprenez le mieux :

Code : PHP

```
<?php
$valeur=end(explode("=", $ligne, 2));
$valeur=substr($ligne, strpos($ligne, "="));
$valeur=preg_replace('#^[^=]+=#', '', $ligne);
?>
```

Bref, explication du code :

On lit chaque ligne du fichier : le premier preg_match() détecte la présence d'un groupe. Si c'en est un, on stocke sa valeur dans \$groupe. \$groupe contient donc toujours le groupe en cours de lecture.

La condition \$ligne[0]!=';' vérifie que l'on n'est pas en train de lire un commentaire (que le premier caractère de la ligne n'est pas un point-virgule).

Et le dernier preg_match() compare le nom de l'item avec celui qui est recherché ; si c'est le cas et que le groupe courant est celui de l'adresse, alors on entre la valeur de l'item dans \$valeur.



😬 Mais... et si ma ligne n'est ni un item ni un groupe, ni un commentaire ?

Eh bien justement, ce qui n'est ni un commentaire ni un groupe **EST un item**.

Exemple : la ligne **ressource>52**.

Ici, on considère que le nom de l'item est "ressource>52" et que la valeur est "" (une chaîne vide).

Retenez bien ceci :

- une ligne qui commence par [et se termine par] est une ouverture de groupe ;
- une ligne commençant par ; est un commentaire ;

- toutes les autres lignes sont des entrées item/valeur.

Afficher le résultat

On peut terminer le code en affichant la valeur trouvée s'il y en a une, et un message d'erreur s'il n'y en a pas :

Code : PHP

```
<?php
if($valeur===false)
    echo "Groupe ou item inexistant";
else
    echo "Valeur trouvée : ".$valeur;
?>
```

Le code PHP

Et voilà le code complet :

Code : PHP

```
<?php
$fichier="membres.ini";
$groupe_recherche="Louise";
$item_recherche="code postal";
$valeur=false;
if(file_exists($fichier) && $fichier_lecture=file($fichier))
    foreach($fichier_lecture as $ligne)
    {
        $ligne_propre=trim($ligne);
        if(preg_match("#^\[(.+)\]$#", $ligne_propre, $matches))
            $groupe=$matches[1];
        else
            if($ligne_propre[0]!=';')
                if($groupe==$groupe_recherche)
                    if(strpos($ligne,$item_recherche."")==0)
                        $valeur=end(explode("=", $ligne, 2));
                    elseif($ligne==$item_recherche)
                        $valeur='';
            }
        else
            echo "Le fichier est introuvable ou incompatible<br />";

        if($valeur===false)
            echo "Groupe ou item inexistant";
        else
            echo "Valeur trouvée : ".$valeur;
    }
?>
```

Vous pouvez tester ce code en [cliquant ici](#)

Formulaire de sélection

Certains ont peut-être déjà essayé de rajouter un formulaire pour choisir l'adresse. Et comme au final, je vous ferai faire un *big* formulaire avec toutes les manipulations possibles (🤖 mais non ne vous découragez pas), alors voici comment faire.

Le début du fichier

Code : PHP

```
<?php
if(isset($_POST["envoyer"]))
{
    $fichier=$_POST["fichier"];
    $groupe_recherche=$_POST["groupe"];
}
```

```

$item_recherche=$_POST["item"];
$valeur=false;
if(file_exists($fichier) && $fichier_lecture=file($fichier))
    foreach($fichier_lecture as $ligne)
    {
        $ligne_propre=trim($ligne);
        if(preg_match("#^\[(.+)\]$#", $ligne_propre, $matches))
            $groupe=$matches[1];
        else
            if($ligne_propre[0]!=";")
                if($groupe==$groupe_recherche)
                    if(strpos($ligne,$item_recherche."")==0)
                        $valeur=end(explode(";", $ligne, 2));
                    elseif($ligne==$item_recherche)
                        $valeur='';
            }
        else
            echo "Le fichier est introuvable ou incompatible<br />";

        if($valeur===false)
            echo "Groupe ou item inexistant";
        else
            echo "Valeur trouvée : ".$valeur;
    }
?>

```

La suite du fichier

Code : HTML

```

<form method="post" action="">
    <fieldset><legend>Adresse de la valeur recherchée</legend>
    <label>Fichier : <input type="text" name="fichier" /></label><br />
    <label>Groupe : <input type="text" name="groupe" /></label><br />
    <label>Item : <input type="text" name="item" /></label><br />
    <input type="submit" name="envoyer" value="Rechercher" />
</fieldset>
</form>

```

Pour essayer ce script, [cliquez ici](#)

Vous savez désormais afficher une valeur issue d'un [.ini](#), vous pouvez passer à l'écriture !

Les outils pour l'écriture

Dans ce chapitre, je vais vous donner tous les outils nécessaires pour l'écriture dans un fichier `.ini` de sorte qu'à la deuxième partie de ce tuto, nous puissions commencer à monter le système de lecture / écriture.



Pour tous les Zér0s qui ont suivi les cours de M@teo21, ce chapitre ne sera qu'un rappel : ceux qui ont l'habitude d'écrire, lire et supprimer des fichiers n'ont pas besoin de le lire et ont juste à passer en revue leurs connaissances avec le QCM. 😊

Le chmod



Attention, tous les fichiers ne sont pas toujours modifiables : pour s'en assurer, utilisez le code suivant :

Code : PHP

```
<?php
$filename = "membres.ini"; //Dans le cas de notre fichier ini
if(is_writable($filename))
{
    echo "Le fichier est accessible en écriture";
}
else
{
    echo "Le fichier n'est pas accessible en écriture";
}
?>
```

Si jamais il n'était pas accessible en écriture, il faudrait alors modifier son chmod. Le chmod est un code donnant accès en lecture, écriture et exécution au propriétaire, au groupe et au public, je n'entre pas dans les détails, voici le code pour le modifier si nécessaire :

Code : PHP

```
<?php
if(chmod ("membres.ini", 0644)) // Toujours donner 4 chiffres car la
notation du mode est octale (et donc aucun chiffre supérieur à 7
dans cet entier).
{
    echo "Le fichier est désormais lisible par tous et modifiable
par le propriétaire.";
}
else
{
    echo "Le changement de chmod a échoué.";
}
?>
```

Si le changement échoue, c'est que votre serveur ne le permet pas ou que le chemin d'accès du fichier est protégé.

Votre fichier est maintenant -normalement- prêt à être modifié ! 😊

Les fonctions de manipulation de fichiers

Tout d'abord, voici comment on procède pour écrire dans un fichier : on ouvre le fichier, on le modifie et on le ferme, exactement comme avec le Bloc-notes. On va donc d'abord voir les fonctions d'ouverture et de fermeture d'un fichier.

fopen() et fclose()

fopen()

fopen() s'emploie de la façon suivante :

Code : PHP

```
<?php
$handle = fopen("membres.ini", "a+");
?>
```

Et que vaut la variable \$handle, me direz-vous ? Eh bien c'est une **ressource** de type stream (flux), autrement dit on ne peut pas l'afficher, on va juste s'en servir en la passant en argument des autres fonctions de manipulation de fichier. Pour en finir avec cette fonction fopen(), le premier argument est donc le nom du fichier et le deuxième est le mode.

Voici les modes les plus courants :

Lettre	Lecture	Écriture	Curseur	Spécial
r	oui	non	au début	-
r+	oui	oui	au début	-
w	non	oui	au début	Efface le contenu du fichier
w+	oui	oui	au début	Efface le contenu du fichier Crée le fichier s'il n'existe pas
a	non	oui	à la fin	Crée le fichier s'il n'existe pas
a+	oui	oui	à la fin	Crée le fichier s'il n'existe pas

On a maintenant ouvert notre fichier, et vous devez savoir qu'il est possible d'ouvrir plusieurs fichiers en même temps en donnant différents noms à vos ressources, vous pouvez ainsi manipuler les fichiers ensemble.

fclose()

Une fois toutes les opérations faites, on ferme le fichier ainsi :

Code : PHP

```
<?php
fclose($handle);
?>
```

fwrite()

Tout d'abord, sachez que cette fonction a un alias : fputs(), c'est-à-dire que vous pouvez utiliser l'une ou l'autre de la même façon ; moi, je préfère *écrire* plutôt que *mettre* (cf. nom de ces fonctions en français). Cette fonction prend en premier paramètre la *flux ressource* et en second, la *chaîne de caractères* à insérer et elle renvoie le nombre d'octets écrits ou **false** en cas d'échec.

Code : PHP

```
<?php
$mon_texte = "[groupe]
item=valeur";
if(fwrite($handle, $mon_texte) === FALSE)
{
    echo "L'écriture a échoué.";
}
?>
```

file_put_contents()

Cette fonction n'existe qu'à partir de PHP 5, si votre version de PHP est plus ancienne, utilisez le premier code ci-dessous.

Et enfin `file_put_contents()` qui est l'équivalent de :

Code : PHP

```
<?php
$chemin = "membres.ini";
$mon_texte = "[groupe]
item=valeur";
$handle = fopen($chemin, "w+");
if(fwrite($handle, $mon_texte) === FALSE)
{
    echo "L'écriture a échoué.";
}
fclose($handle);
?>
```

Et avec `file_put_contents()`, il suffit de faire ça :

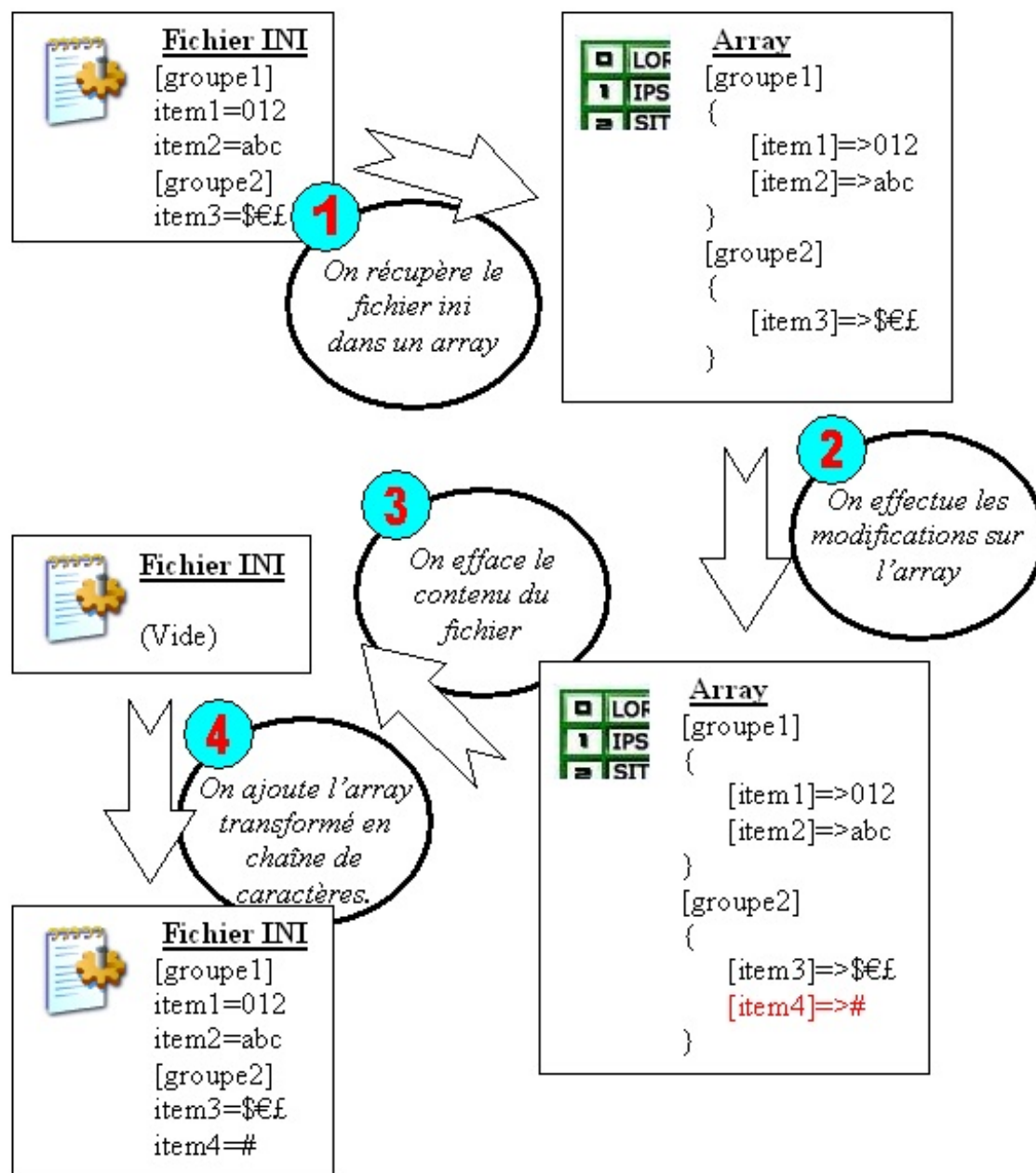
Code : PHP

```
<?php
$chemin = "membres.ini";
$mon_texte = "[groupe]
item=valeur";
if(file_put_contents($chemin, $mon_texte) === FALSE)
{
    echo "Echec de l'écriture.";
}
?>
```

Attention : cette fonction magique a ses limites, car elle agit comme si on avait entré le mode `w` ou `w+` à `fopen()`. Vous êtes donc obligés de réutiliser les trois autres fonctions si vous voulez entrer un autre mode. De plus, cette fonction n'existe que depuis PHP 5, elle n'est donc pas disponible sur tous les hébergeurs. Certains encore exigent une extension [.php5](#) des fichiers ; si vous n'êtes pas sûrs, utilisez plutôt la première méthode.

Effacer pour mieux réécrire

C'est le principe de la modification : on récupère le contenu d'un fichier, on le change puis on remplace l'ancien contenu par le nouveau. Voilà le schéma de l'opération en ce qui concerne le [.ini](#) :



Et là-dessus, on sait tout faire sauf transformer un array en texte pour fichier **.ini** (pas de panique : on le verra dans le chapitre suivant). La dernière chose dont vous avez besoin avant qu'on finisse ce chapitre, c'est de méthodes pour effacer le contenu d'un fichier.

Pour ça, vous avez le mode **w** (et **w+**) de **fopen()**, ou la fonction **file_put_contents()** qui efface le fichier avant d'écrire dedans. Enfin, la dernière méthode consiste à supprimer le fichier avec la fonction **unlink()**.

Code : PHP

```
<?php
$filename = "membres.ini";
if(unlink($filename))
{
    echo "[" . $filename . "] a été supprimé.";
}
else
{
    echo "La suppression de [" . $filename . "] a échoué.";
}
?>
```

Mais si le fichier à supprimer n'existe pas, **unlink()** renvoie une erreur : donc, pour éviter cette tache dans la page, on vérifie que le fichier existe avec **file_exists()**.

Code : PHP

```
<?php
if(file_exists($filename))
{
    echo "Le fichier existe.";
}
else
{
    echo "Le fichier n'existe pas.";
}
?>
```

Donc si le fichier n'est pas protégé, il suffit de mettre cette ligne de code pour le supprimer :

Code : PHP

```
<?php
$filename="membres.ini";
//Ici, on ne sait si le fichier $filename existe
if(file_exists($filename)) unlink($filename);
//Ici, on n'a aucun message d'erreur et on est sûrs que le fichier
$filename n'existe plus
?>
```

Après le QCM, vous serez prêts à créer le système complet de gestion des fichiers [.ini](#).


Et voilà, vous avez maintenant tout ce qu'il vous faut : nous allons pouvoir créer notre page de gestion des fichiers [.ini](#).

Voilà, vous êtes arrivés à la moitié de ce tuto ; désormais, il va y avoir beaucoup plus de pratique. 😊

Partie 2 : Une page de gestion des fichiers .ini

Cette partie aura pour but de vous faire construire morceau par morceau une page de gestion des fichiers .ini complète sur votre site web.

Transformer un array

Nous allons maintenant voir comment récupérer un array depuis un fichier .ini et inversement (en utilisant ce qu'on a vu précédemment); il nous sera utile de savoir modifier un array, vous savez bien entendu comment faire...  Bon : pour ceux qui auraient un doute, voici les trois opérations qu'on peut lui faire subir.

Changer une valeur

Code : PHP

```
<?php
$groupe = "Louise";
$item = "age";
$array[$groupe][$item] = $valeur;
?>
```

Ajouter un item ou un groupe

Comme avant, sauf qu'il suffit de prendre pour groupe et item des noms qui n'y sont pas encore.

Supprimer un item ou un groupe

On supprime une entrée dans un array comme n'importe quelle autre variable avec unset() :

Code : PHP

```
<?php
unset($array[$groupe][$item]); //Supprime l'item
unset($array[$groupe]); //Supprime le groupe
?>
```

Avec ça, vous êtes prêts pour ce chapitre.

Du fichier à l'array

Pour récupérer le fichier entier dans un array, nous allons utiliser le code de la première partie en le changeant à peine :

Code : PHP

```
<?php
$fichier="membres.ini";
$array=array();
if(file_exists($fichier) && $fichier_lecture=file($fichier))
    foreach($fichier_lecture as $ligne)
    {
        if(preg_match("#^\[(.*)\]\s+$#", $ligne, $matches))
        {
            $groupe=$matches[1];
            $array[$groupe]=array();
        }
        elseif($ligne[0]!=';')
        {
            list($item,$valeur)=explode("=", $ligne, 2);
            if(!isset($valeur)) // S'il n'y a pas de valeur
                $valeur=''; // On donne une chaîne vide
            $array[$groupe][$item]=$valeur;
        }
    }
}
```



```

    }
}
else
echo "Le fichier est introuvable ou incompatible<br />";
// Et voilà, $array contient le fichier membres.ini sous la forme
d'un array à 2 niveaux.
?>

```

Reportez-vous au premier chapitre expliquant comment récupérer une valeur ; seule différence ici : on rentre les groupes et items dans \$array.

De l'array au fichier

Voici donc la manipulation inverse dans laquelle on va également se servir de foreach, mais en utilisant en plus la clé, par exemple, pour un array structuré comme ça :

Code : Autre

```

$array
{
    [$key] => $value
    [$key] => $value
    [$key] => $value
    [$key] => $value
}

```

On peut parcourir cet array avec

`foreach($array as $key => $value)`

souvent utile quand on a un array associatif.

Code : PHP

```

<?php
$fichier_save="";
foreach($array as $key => $groupe_n)
{
    $fichier_save.="\\n[".$key."]"; // \\n est une entrée à la ligne
    foreach($groupe_n as $key => $item_n)
    {
        $fichier_save.="\\n".$key."="."$item_n";
    }
}
$fichier_save=substr($fichier_save, 1); // On enlève le premier
caractère qui est -si vous regardez bien- forcément une entrée à la
ligne inutile
if(false===@file_put_contents("membres.ini", $fichier_save))
{
    echo "Impossible d'écrire dans ce fichier";
}
?>

```

Vous savez maintenant comment récupérer un array, comment le modifier et sauver les modifications ! 😊

Un objet pour tout rassembler

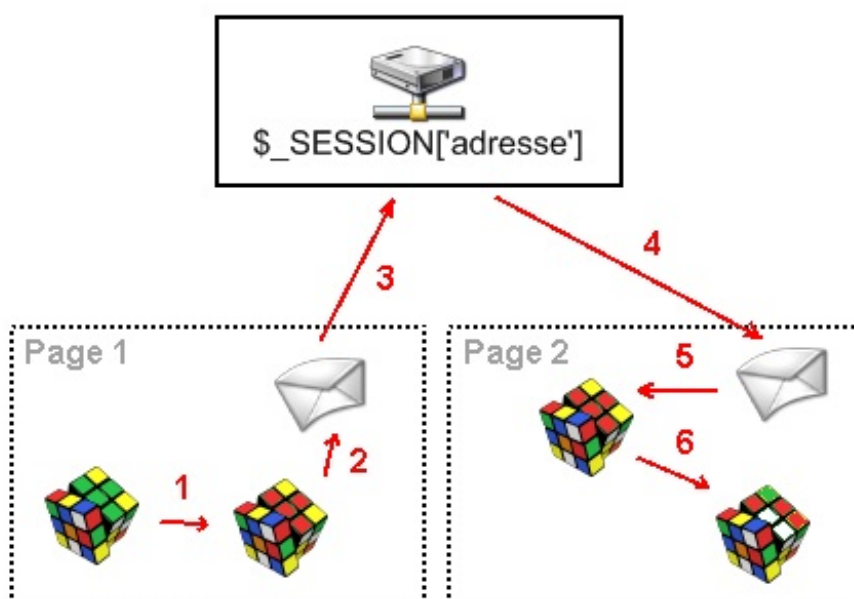
Vous avez maintenant un bon nombre de morceaux de code PHP pour les différentes opérations qu'on peut être amenés à faire. Et je vous propose, pour les organiser, de les ranger tous ensemble dans un objet. Pas de panique, nous allons tout de suite voir ce que c'est.

Qu'est-ce qu'un objet ?

Un objet est en gros une boîte où l'on peut ranger des **variables** et des **fonctions** (fonctions qui seront appelées **méthodes**). L'avantage de l'objet est premièrement d'organiser le code et ensuite, de pouvoir être transformé en chaîne de caractères pour être inséré dans une variable. Cette propriété est très intéressante lorsqu'on voudra transmettre notre objet (avec toutes ses variables) dans une variable de SESSION.

Et c'est là que ça devient intéressant pour nous car nous allons pouvoir conserver les variables dans lesquelles on enregistre les noms des fichier, groupe et item sélectionnés. Fouiller un fichier .ini pourra se faire en plusieurs chargements de pages. On pourra, par exemple, modifier l'item de notre adresse sans donner à nouveau les informations du groupe et du fichier.

Petit schéma de cette technique :



Légende des flèches rouges

1. On modifie notre objet.
2. On transforme notre objet en chaîne.
3. On stocke la chaîne dans la variable de session 'adresse'.
4. Sur une autre page, on récupère cette variable.
5. On extrait la chaîne pour recréer l'objet.
6. On peut modifier une ou plusieurs variables de notre objet et recommencer l'opération.

Créer un objet

Pour créer un objet, il faut précéder le nom de l'objet de **class** :

Code : PHP

```
<?php
class objet
{
    // Méthodes et variables
}
?>
```

L'objet objet vient d'être déclaré et tout comme les fonctions déclarées, il faut encore les appeler pour pouvoir les utiliser.

Code : PHP

```
<?php
$obj = new objet();
?>
```

Et enfin pour appeler une méthode ou une variable de \$obj, on utilise la flèche (->), exemple :

Code : PHP

```
<?php
echo $obj->methode();
echo $obj->variable;
?>
```

Pour créer des variables dans votre objet, il faut utiliser le mot-clé **var** :

Code : PHP

```
<?php
class objet
{
    var $variable;
}
?>
```

Et vous pouvez ensuite lui assigner une valeur et la manipuler comme une variable normale, sauf qu'on rajoute **this->** aux noms des variables :

Code : PHP

```
<?php
class objet
{
    var $variable1;
    $this->variable1="Bla";
    $this->variable1.=" bla";
    var $variable2=", stop !";
}
?>
```

Ici, si on utilisait ensuite ce code :

Code : PHP

```
<?php
$obj = new objet();
echo $obj->$variable1.$obj->$variable2;
?>
```

cela afficherait : Bla bla, stop !

Et enfin, pour créer une méthode :

Code : PHP

```
<?php
class objet
{
    function methode()
    {
        // contenu de la méthode
    }
}
```

>

Transmettre dans une SESSION

Code : PHP

```
<?php
class objet()
{
    // On crée l'objet
}
session_start();
if(isset($_SESSION["objet"]))
$ini_objet=unserialize($_SESSION["objet"]);
else
$ini_objet=new objet();

// Ici le contenu de la page

// Et en fin de page
$_SESSION["objet"]=serialize($ini_objet);
?>
```

Donc on regarde s'il y a quelque chose dans la variable de session : si oui, on crée un objet avec, sinon on crée un objet vierge. On le manipule tout au long de la page et à la fin, on le sauvegarde dans la variable de session. Les fonctions `serialize()` et `unserialize()` servent respectivement à transformer un objet en chaîne de caractères et inversement, à le recréer.

Pour apprendre à se servir des objets

J'ai passé en revue ce dont on aura besoin pour notre page, mais sachez qu'on peut faire plein de choses avec les objets ; voici des liens pour approfondir vos connaissances dans ce domaine :

- [sur le Site du Zéro](#) ;
- [sur Commentçamarche.net](#) ;
- [sur PHP.Développez.com](#).

Nous allons à présent pouvoir créer un objet sur mesure adapté à nos besoins !

Notre objet INI

Nous allons maintenant établir le rôle de chacune des méthodes dont nous aurons besoin et je vous fournirai le code PHP correspondant. Dans la dernière partie, je propose une organisation en héritage pour trier les méthodes par genre.

Les méthodes une à une

Voici d'abord nos variables :

\$fichier, \$groupe, \$item, \$valeur, \$fichier_ini. Les 3 premières sont respectivement les noms du fichier, du groupe et de l'item, \$valeur est la valeur courante à la position donnée par les coordonnées et \$fichier_ini est l'array contenant tout le .ini :

Code : PHP

```
<?php
class ini
{
    var $fichier="";
    var $groupe="";
    var $item="";
    var $valeur="";
    var $fichier_ini=array();
```

m_fichier()

Cette méthode reçoit en paramètre l'adresse du fichier à lire et elle transforme le fichier en array qu'il stocke dans \$fichier_ini :

Code : PHP

```
<?php
function m_fichier($arg)
{
    $this->fichier=$arg;
    $this->fichier_ini=null;
    $this->fichier_ini=array();
    // Pour utiliser parse_ini_file() par défaut, enlevez /* et */,
    sinon supprimez ce commentaire (conseillé)
    /*if(false!==( $array=@parse_ini_file($arg, TRUE)))
    {
        $this->fichier_ini=$array;
    }
    else*/if(file_exists($arg) && $fichier_lecture=file($arg))
    {
        foreach($fichier_lecture as $ligne)
        {
            $ligne_propre=trim($ligne);
            if(preg_match("#^\[(.+)\]$#", $ligne_propre, $matches))
            {
                $groupe_curseur=$matches[1];
            }
            else
            {
                if($ligne_propre[0]!=';' &&
                $tableau=explode("=", $ligne, 2))
                {
                    $this->fichier_ini[$groupe_curseur][$tableau[0]]=rtrim($tableau[1], "\n\r");
                    // rtrim nettoie les caractères \n et \r en fin de
                    chaîne,
                    // cela a pour effet d'enlever l'entrée à la ligne qui
                    la termine.
                }
            }
        }
        $this->valeur=$this->fichier_ini[$this->groupe][$this->item];
    }
```

```
?>
```

J'en profite pour vous parler de la fonction `parse_ini_file()` qui fait cette opération que je vous conseille de ne pas utiliser, car elle ne récupère pas l'intégralité d'un item : s'il trouve le signe '=' dans une valeur, la fonction ne renverra que ce qu'il y a avant. Cela est rarement un avantage et si jamais vous vouliez uniquement récupérer ce qu'il y a avant le premier '=' d'une valeur, il y a un moyen simple de le faire :

Code : PHP

```
<?php $valeur_avant_egal=reset(explode("=", $valeur_initiale)); ?>
```

De plus la fonction `parse_ini_file()` est à usage unique alors que la fonction `m_fichier` (ci-dessus) est adaptable, c'est à dire que vous pouvez décider que, chez vous, les commentaires commencent par # et non par ; ou encore que votre array a un niveau de plus et donc que le fichier contient des sous-groupe délimités par {groupe} ou <groupe>. Et enfin dans certains cas, le type des données stockées peut être important. Alors des préfixes comme **i**, **f**, **s** : peuvent indiquer si la valeur est un nombre entier, décimal ou une chaîne de caractères. Bref, le stockage d'un array dans un fichier ne se limite pas à la technique des fichiers .ini, il n'a de limite que votre imagination. 😊

m_groupe()

Pour sélectionner un groupe :

Code : PHP

```
<?php
function m_groupe($arg)
{
    $this->groupe=$arg;
    $this->valeur=$this->fichier_ini[$this->groupe][$this->item];
    return true;
}
?>
```

m_item()

Pour sélectionner un item :

Code : PHP

```
<?php
function m_item($arg)
{
    $this->item=$arg;
    $this->valeur=$this->fichier_ini[$this->groupe][$this->item];
    return true;
}
?>
```

m_put()

Pour changer une valeur. Avec 1 paramètre, la valeur est insérée dans l'item sélectionné et le groupe sélectionné. Les paramètres suivants sont optionnels : il s'agit dans l'ordre de l'item, du groupe et du fichier qui sont sélectionnés si les paramètres correspondants sont fournis :

Code : PHP

```
<?php
function m_put($arg, $arg_i=false, $arg_g=false, $arg_f=false)
{
    if($arg_f!==false) $this->m_fichier($arg_f);
    if($arg_g!==false) $this->m_groupe($arg_g);
    if($arg_i!==false) $this->m_item($arg_i);
    $this->fichier_ini[$this->groupe][$this->item]=$arg;
    $this->valeur=$arg;
    return $this->fichier." ==> [".$this->groupe."] ".$this->
    >item."=".$this->valeur;
}
?>
```

m_count()

Si aucun paramètre n'est fourni, m_count renvoie un array contenant à l'entrée 0 le nombre total d'éléments dans le fichier .ini, à l'entrée 1 le nombre de groupes dans le fichier et à l'entrée 2 le nombre d'items.

Si un paramètre est fourni, m_count renvoie le nombre d'items dans le groupe fourni en paramètre.

Code : PHP

```
<?php
function m_count($arg_gr=false)
{
    if($arg_gr===false)
        return array(1=>$gr_cou=count($this->fichier_ini),
0=>$itgr_cou=count($this->fichier_ini, COUNT_RECURSIVE),
2=>$itgr_cou-$gr_cou);
    else
        return count($this->fichier_ini[$arg_gr]);
}
?>
```

array_groupe()

Si un paramètre est fourni, array_groupe renvoie un array associatif contenant les items du groupe fourni en paramètre :

Code : PHP

```
<?php
function array_groupe($arg_gr=false)
{
    if($arg_gr===false)
        $arg_gr=$this->groupe;
    return $this->fichier_ini[$arg_gr];
}
?>
```

À noter : les deux lignes ci-dessous sont équivalentes :

Code : PHP

```
<?php
print_r($objet->array_groupe('mon groupe'));
print_r($objet->fichier_ini['mon groupe']);
?>
```

save()

C'est l'opération inverse de `m_fichier`, cette fonction remplace le contenu du fichier par l'array en cours de traitement. `save()` sauvegarde donc les modifications apportées à 1 fichier :

Code : PHP

```
<?php
function save()
{
    $fichier_save="";
    foreach($this->fichier_ini as $key => $groupe_n)
    {
        $fichier_save.="
[". $key. "];
        foreach($groupe_n as $key => $item_n)
        {
            $fichier_save.="
". $key. "=" . $item_n;
        }
    }
    $fichier_save=substr($fichier_save, 1);
    if(file_exists($this->fichier) &&
reset(explode('.',phpversion()))>=5)
    {
        if(false===file_put_contents($this->fichier,
$fichier_save))
        {
            die("Impossible d'écriture dans ce fichier
(mais le fichier existe).");
        }
    }
    else
    {
        $fichier_ouv=fopen($this->fichier,"w+");
        if(false===fwrite($fichier_ouv, $fichier_save))
        {
            die("Impossible d'écriture dans ce fichier (Le
fichier n'existe pas).");
        }
        fclose($fichier_ouv);
    }
    return true;
}
?>
```

À noter : il faut utiliser cette fonction avant de changer de fichier pour enregistrer les modifications. Peut-être que certains d'entre vous se disent qu'il vaudrait mieux inclure cette fonction dans `m_put` pour éviter de l'oublier et éviter des lignes de codes inutiles, mais il peut s'avérer utile (j'en ai eu besoin) de modifier l'objet sans modifier le fichier.

clear()

Réinitialise toutes les variables de l'objet et efface l'array `$fichier_ini`.

Code : PHP

```
<?php
function clear()
{
    $this->fichier="";
    $this->groupe="";
    $this->item="";
    $this->valeur="";
    $this->fichier_ini=null;
    $this->fichier_ini=array();
}
```



```
}  
?>
```

s_fichier()

Supprime le fichier en cours d'exécution s'il existe.

Code : PHP

```
<?php  
function s_fichier()  
{  
    $return=$this->fichier;  
    if(file_exists($this->fichier)) unlink($this->fichier);  
    $this->fichier="";  
    $this->valeur="";  
    return "fichier(".$return.") supprimé;";  
}  
?>
```

s_groupe()

Supprime le groupe sélectionné.

Code : PHP

```
<?php  
function s_groupe()  
{  
    $return=$this->groupe;  
    if(isset($this->fichier_ini[$this->groupe])) unset($this->  
fichier_ini[$this->groupe]);  
    $this->groupe="";  
    $this->valeur="";  
    return "groupe(".$return.") supprimé;";  
}  
?>
```

s_item()

Supprime l'item sélectionné.

Code : PHP

```
<?php  
function s_item()  
{  
    $return=$this->item;  
    if(isset($this->fichier_ini[$this->groupe][$this->item]))  
unset($this->fichier_ini[$this->groupe][$this->item]);  
    $this->item="";  
    $this->valeur="";  
    return "item(".$return.") supprimé;";  
}  
?>
```

print_curseur()

Affiche les coordonnées du curseur et la valeur courante.

Code : PHP

```
<?php
function print_curseur()
{
    echo "Fichier : <b>".$this->fichier."</b><br />";
    echo "Groupe : <b>".$this->groupe."</b><br />";
    echo "Item : <b>".$this->item."</b><br />";
    echo "Valeur : <b>".$this->valeur."</b><br />";
    return true;
}
?>
```

print_dossier()

Seulement si \$fichier contient l'adresse d'un dossier.

Pour fonctionner correctement, il vous faut les images suivantes :



Cette fonction ne reçoit aucun paramètre et affiche un arbre contenant au sommet l'adresse entrée dans la méthode m_fichier() et en dessous, la liste des fichiers .ini que contient ce dossier (utile surtout dans la page d'administration).

Code : PHP

```
<?php
function print_dossier()
{
    if(is_dir($this->fichier)) {
        echo "<img src='dir.png' alt='Dossier' /><span
style='position:relative; top:-10px;font-size:20px; font-
weight:bold;'>".$this->fichier."</span><br />";
        if($handle=opendir($this->fichier))
        {
            while(false!=($file=readdir($handle)))
            {
                if(substr($file, -4, 4)==".ini")
                {
                    echo "&nbsp;&nbsp;&nbsp;<a href='?fichier=".$file."'><img
src='iniicone.png' alt='Ini' style='border:none;'
/></a>&nbsp;&nbsp;&nbsp;".$file."<br />";
                }
            }
            closedir($handle);
        }
        return true; }
    else { echo "L'élément sélectionné n'est pas un dossier";
return false; }
}
?>
```

print_fichier()

Affiche un aperçu du fichier et surligne le groupe et l'item sélectionné.

Code : PHP

```

<?php
function print_fichier()
{
    if(file_exists($this->fichier) && is_file($this->fichier) &&
    $fichier_lecture=file($this->fichier))
    {
        foreach($fichier_lecture as $ligne)
        {
            $ligne=preg_replace("#\s$#", "", $ligne);
            if(preg_match("#^\[.+\]\s?$#", $ligne))
                $groupe=false;
            if(preg_match("#^\[" . preg_quote($this->groupe,
            "#") . "\]$#", $ligne))
            {
                $echo.= "<span style='background-
color:aqua;'>".htmlspecialchars($ligne). "</span><br />";
                $groupe=true;
            }
            elseif($groupe==true && preg_match("#^(" . $this->
            item."=#", $ligne))
                $echo.= "<span style='background-
color:yellow;'>".htmlspecialchars($ligne). "</span><br />";
            elseif($groupe==true && $this->
            item==reset(explode("=", $ligne)))
                $echo.= "<span style='background-
color:yellow;'>".htmlspecialchars($ligne). "</span><br />";
            else
                $echo.= htmlspecialchars($ligne). "<br />";
        }
        echo $echo;
    }
    else
    {
        echo "Le fichier n'existe pas ou est incompatible";
    }
    $this->valeur=$this->fichier_ini[$this->groupe][$this->item];
    return true;
}
?>

```

m_valeur()

Prend deux paramètres : le premier est l'item et le deuxième le groupe. m_valeur() renvoie la valeur (sous forme d'une chaîne de caractères) qui se trouve à l'item donné du groupe donné dans le fichier en cours de traitement.

Code : PHP

```

<?php
function m_valeur($arg_item, $arg_groupe)
{
    return $this->fichier_ini[$arg_groupe][$arg_item];
}
?>

```

On termine bien sûr par fermer la déclaration de class avec }.

Le code PHP

Code : PHP

```

<?php
class ini
{
    var $fichier="";

```

```

var $groupe="";
var $item="";
var $valeur="";
var $fichier_ini=array();
function m_fichier($arg)
{
    $this->fichier=$arg;
    $this->fichier_ini=null;
    $this->fichier_ini=array();
    // Pour utiliser parse_ini_file() par défaut, enlevez /* et */,
    sinon supprimez ce commentaire (conseillé)
    /*if(false!==( $array=@parse_ini_file($arg, TRUE)))
    {
    $this->fichier_ini=$array;
    }
else*/if(file_exists($arg) && $fichier_lecture=file($arg))
    {
        foreach($fichier_lecture as $ligne)
        {
            $ligne_propre=trim($ligne);
            if(preg_match("#^\[(.+)\]$", $ligne_propre, $matches))
            {
                $groupe_curseur=$matches[1];
            }
            else
            {
                if($ligne_propre[0]!=';' &&
$tableau=explode("=", $ligne, 2))
                {
                    $this->fichier_ini[$groupe_curseur][$tableau[0]]=rtrim($tableau[1], "\n\r");
                }
            }
        }
        $this->valeur=$this->fichier_ini[$this->groupe][$this->item];
    }
function m_groupe($arg)
{
    $this->groupe=$arg;
    $this->valeur=$this->fichier_ini[$this->groupe][$this->item];
    return true;
}
function m_item($arg)
{
    $this->item=$arg;
    $this->valeur=$this->fichier_ini[$this->groupe][$this->item];
    return true;
}
function m_put($arg, $arg_i=false, $arg_g=false, $arg_f=false)
{
    if($arg_f==false) $this->m_fichier($arg_f);
    if($arg_g==false) $this->m_groupe($arg_g);
    if($arg_i==false) $this->m_item($arg_i);
    $this->fichier_ini[$this->groupe][$this->item]=$arg;
    $this->valeur=$arg;
    return $this->fichier." ==> [".$this->groupe."] ".$this->item."=".$this->valeur;
}
function m_count($arg_gr=false)
{
    if($arg_gr===false)
        return array(1=>$gr_cou=count($this->fichier_ini),
0=>$itgr_cou=count($this->fichier_ini, COUNT_RECURSIVE),
2=>$itgr_cou-$gr_cou);
    else
        return count($this->fichier_ini[$arg_gr]);
}
function array_groupe($arg_gr=false)
{

```

```

        if($arg_gr===false)
        $arg_gr=$this->groupe;
        return $this->fichier_ini[$arg_gr];
    }
    function save()
    {
        $fichier_save="";
        foreach($this->fichier_ini as $key => $groupe_n)
        {
            $fichier_save.="
[". $key. "];
            foreach($groupe_n as $key => $item_n)
            {
                $fichier_save.="
". $key. "=". $item_n;
            }
            $fichier_save=substr($fichier_save, 1);
            if(file_exists($this->fichier) &&
reset(explode('.',phpversion()))>=5)
            {
                if(false===file_put_contents($this->fichier,
$fichier_save))
                {
                    die("Impossible d'&eacute;crire dans ce fichier (mais
le fichier existe).");
                }
            }
            else
            {
                $fichier_ouv=fopen($this->fichier,"w+");
                if(false===fwrite($fichier_ouv, $fichier_save))
                {
                    die("Impossible d'&eacute;crire dans ce fichier (Le
fichier n'existe pas).");
                }
                fclose($fichier_ouv);
            }
            return true;
        }
    }
    function clear()
    {
        $this->fichier="";
        $this->groupe="";
        $this->item="";
        $this->valeur="";
        $this->fichier_ini=null;
        $this->fichier_ini=array();
    }
    function s_fichier()
    {
        $return=$this->fichier;
        if(file_exists($this->fichier)) unlink($this->fichier);
        $this->fichier="";
        $this->valeur="";
        return "fichier(".$return.") supprim&eacute;.";
    }
    function s_groupe()
    {
        $return=$this->groupe;
        if(isset($this->fichier_ini[$this->groupe])) unset($this-
>fichier_ini[$this->groupe]);
        $this->groupe="";
        $this->valeur="";
        return "groupe(".$return.") supprim&eacute;.";
    }
    function s_item()
    {
        $return=$this->item;
        if(isset($this->fichier_ini[$this->groupe][$this->item]))

```

```

unset($this->fichier_ini[$this->groupe][$this->item]);
$this->item="";
$this->valeur="";
return "item(".$return.") supprimé.";
}
function print_curseur()
{
    echo "Fichier : <b>".$this->fichier."</b><br />";
    echo "Groupe : <b>".$this->groupe."</b><br />";
    echo "Item : <b>".$this->item."</b><br />";
    echo "Valeur : <b>".$this->valeur."</b><br />";
    return true;
}
function print_dossier()
{
    if(is_dir($this->fichier)) {
        echo "<img src='dir.png' alt='Dossier' /><span
style='position:relative; top:-10px;font-size:20px; font-
weight:bold;'>".$this->fichier."</span><br />";
        if($handle=opendir($this->fichier))
        {
            while(false!=( $file=readdir($handle)))
            {
                if(substr($file, -4, 4)==".ini")
                {
                    echo "&nbsp;&nbsp;&nbsp;<a href='?fichier=".$file."'><img
src='iniicone.png' alt='Ini' style='border:none;'
/></a>&nbsp;&nbsp;".$file."<br />";
                }
            }
            closedir($handle);
        }
        return true; }
    else { echo "L'élément sélectionné n'est pas un dossier"; return
false; }
}
function print_fichier()
{
    if(file_exists($this->fichier) && is_file($this->fichier) &&
$fichier_lecture=file($this->fichier))
    {
        foreach($fichier_lecture as $ligne)
        {
            $ligne=preg_replace("#\s$#", "", $ligne);
            if(preg_match("#^\[.+]\s?$#", $ligne))
            $groupe=false;
            if(preg_match("#^\[\".preg_quote($this->groupe,
\"#\").\"\\]$#", $ligne))
            {
                $echo.= "<span style='background-
color:aqua;'>".htmlspecialchars($ligne)."</span><br />";
                $groupe=true;
            }
            elseif($groupe==true && $this-
>item==reset(explode("=", $ligne)))
                $echo.= "<span style='background-
color:yellow;'>".htmlspecialchars($ligne)."</span><br />";
            else
                $echo.= htmlspecialchars($ligne)."<br />";
        }
        echo $echo;
    }
    else
    {
        echo "Le fichier n'existe pas ou est incompatible";
    }
    $this->valeur=$this->fichier_ini[$this->groupe][$this->item];
    return true;
}
function m valeur($arg item, $arg groupe)

```

```
{  
    return $this->fichier_ini[$arg_groupe][$arg_item];  
}  
}  
?>
```

Trier les méthodes grâce à l'héritage

Grâce au mot-clé `extends` nous allons pouvoir créer plusieurs class complémentaires. Je vous propose de faire ainsi : ne laissez dans la class `ini` que les méthodes de lectures :

Code : PHP

```
<?php  
class ini  
{  
    var $fichier="";  
    var $groupe="";  
    var $item="";  
    var $valeur="";  
    var $fichier_ini=array();  
    function m_fichier( ...  
    function m_groupe( ...  
    function m_item( ...  
    function m_valeur( ...  
    function m_count( ...  
}  
?>
```

Voilà donc la class utile pour lire un fichier `.ini`, mettez ça dans un fichier que vous incluez avec `include()` ou `require()` dans votre fichier `.php`. Puis dans un autre fichier, mettez les méthodes d'écriture :

Code : PHP

```
<?php  
class ini_e extends ini  
{  
    function m_put( ...  
    function save( ...  
    function s_fichier( ...  
    function s_groupe( ...  
    function s_item( ...  
}  
?>
```

que vous n'incluez dans vos pages que si vous avez besoin de modifier un fichier `.ini` : vous appellerez alors `ini_e` et non `ini` pour créer votre objet. Vous pouvez enfin faire une extension de `ini_e` avec les méthodes `print_` et celles qui restent. Ce qui permet de ne pas déclarer des méthodes dont on n'a pas besoin dans une page ou une autre.

La partie héritage, c'est du bonus : ça n'est ni vraiment essentiel pour le traitement INI, ni utile pour la suite du tuto.

Le formulaire

Il ne reste que le plus simple !! Un bête formulaire et son traitement qui vous permettront de manipuler les fichiers .ini de votre site, l'équivalent de phpMyAdmin pour les bases de données SQL.



Des erreurs de type Notice ont été signalées, cela est dû à un niveau de rapport d'erreur un peu trop élevé. Si des erreurs Notice s'affichent chez vous, ajoutez simplement en première ligne du code **error_reporting(E_ALL ^ E_NOTICE);**

Le HTML

Pour ne pas parler dans le vide, voici à quoi ressemble ma page d'administration : simple mais efficace.

Fichier Fichier : <input type="text"/> <input type="button" value="Changer"/> <input type="button" value="Supprimer"/>	Information sur le curseur Fichier : ../administration/races.ini 6 éléments dans ce fichier Groupe : 2 dont 3 groupes et 3 items. Item : nom 1 items dans ce groupe. Valeur : Nautolan
Groupe Groupe : <input type="text"/> <input type="button" value="Changer"/> <input type="button" value="Supprimer"/>	Aperçu du fichier [1] nom=Humain [2] nom=Nautolan [3] nom=Twilek
Item Item : <input type="text"/> <input type="button" value="Changer"/> <input type="button" value="Supprimer"/>	
Valeur Valeur : <input type="text" value="Zabrak"/> <input type="button" value="Changer"/> <input type="button" value="Supprimer"/>	
Réinitialiser <input type="button" value="Effacer"/>	

Et dans Aperçu du fichier s'utilise la fonction print_dossier que certains n'ont pas dû trouver utile, et ça donne ça :

```

.. /administration/
├── multi.ini
├── places.ini
└── races.ini
  
```

Code : PHP

```

<?php
// D'abord on insère notre class
require('./require/class.php');
// Ou avec include
include('./include/class.php');

session_start();
if(isset($_SESSION['ini']))
    $ini_objet=unserialize($_SESSION['ini']);
else
    $ini_objet=new ini();

// Ensuite, on traite le formulaire avec PHP - voire à la suite
/* Insérez le code de traitement du formulaire ICI */

$_SESSION['ini']=serialize($ini_objet);
/* La ligne précédente et les 5 plus haut permettent de sauvegarder
les opérations réalisées dans une variable de session et donc, de
pouvoir circuler dans un .ini en plusieurs chargements de page. */
  
```


>

Code : PHP

```

<html>
<head>
<title>Traitement INI</title>
<style>
form
{
    margin:0px;
    padding:0px;
}
fieldset
{
    margin-right:10px;
    margin-bottom:8px;
    border:1px black solid;
}
legend
{
    color:#333333;
    font-style:italic;
    font-weight:bolder;
    font-size:18px;
}
</style>
</head>
<body>
<div style="float:right;">
    <div style="width:400px; padding:6px; margin-right:12px;
border:1px solid black;">
        <div style="float:right; text-align:right;">
            <?php
                $stat=$ini_objet->m_count();
                echo '<strong>'.$stat[1].'&acute;l&acute;ments</strong> dans
ce fichier<br />
dont <strong>'.$stat[0].'groupes</strong> et <strong>'.$stat[2].'
items</strong>.<br />
<strong>'.$ini_objet->m_count($ini_objet->groupe).' items</strong>
dans ce groupe.';
            <?>
        </div>
        <u><i><b>Information sur le curseur</b></i></u><br />
        <?php $ini_objet->print_curseur(); <?>
        </div>
        <div style="width:400px; height:400px; overflow:auto; padding:6px;
margin-top:12px; margin-right:12px; border:1px solid black;">
            <u><i><b>Aper&ccedil;u du <?php if(is_dir($ini_objet->fichier))
echo 'dossier'; else echo 'fichier'; ?></b></i></u><br /><br />
            <span style="font-family:courier; white-space:nowrap;"><?php
if(is_dir($ini_objet->fichier)) $ini_objet->print_dossier(); else
$ini_objet->print_fichier(); ?></span>
            </div>
        </div>
<fieldset><legend>Fichier</legend>
    <form method="post" action="">
        <label>Fichier : <input type="text" name="source" /></label><br
/>
        <input type="submit" name="fichier" value="Changer" />
    </form>
    <form method="post" action="">
        <input type="submit" name="fichier" value="Supprimer" />
    </form>
</fieldset>
<fieldset><legend>Groupe</legend>
    <form method="post" action="">

```

```

        <label>Groupe : <input type="text" name="source" /></label><br
    />
    <input type="submit" name="groupe" value="Changer" />
</form>
<form method="post" action="">
    <input type="submit" name="groupe" value="Supprimer" />
</form>
</fieldset>
<fieldset><legend>Item</legend>
    <form method="post" action="">
        <label>Item : <input type="text" name="source" /></label><br />
        <input type="submit" name="item" value="Changer" />
    </form>
    <form method="post" action="">
        <input type="submit" name="item" value="Supprimer" />
    </form>
</fieldset>
<fieldset><legend>Valeur</legend>
    <form method="post" action="">
        <label>Valeur : <input type="text" name="source" /></label><br
    />
    <input type="submit" name="valeur" value="Changer" />
</form>
<form method="post" action="">
    <input type="submit" name="valeur" value="Supprimer" />
</form>
</fieldset>
<br />
<fieldset><legend>Réinitialiser</legend>
    <form method="post" action="">
        <input type="submit" name="clear" value="Effacer" />
    </form>
</fieldset>
</body>
</html>

```

Vous venez de découvrir un formulaire on ne peut plus simple avec une balise de style CSS (que vous pouvez extraire dans un fichier .css) pour obtenir le design sommaire de la screen. Pour les cadres de droite, il s'agit de l'affichage avec les fonctions print et count de l'objet ini.

Traitement PHP

J'ai préféré vous donner le formulaire d'abord et le traitement PHP ensuite, tout simplement parce que ça me paraît l'ordre logique. Voici donc le code PHP complet en sachant que dans `./include/class.php` se trouve la class ini que nous avons construite morceau par morceau dans les chapitres précédents, et dont le code complet se trouve juste au chapitre antérieur.

Ce code est à insérer à la place du commentaire `/* Insérez le code de traitement du formulaire ICI */` du code PHP précédent.

Code : PHP

```

<?php
include ('./include/class.php');

session_start();
if(isset($_SESSION['ini']))
    $ini_objet=unserialize($_SESSION['ini']);
else
    $ini_objet=new ini();

if(isset($_GET['fichier']))
{
    $dossier=$ini_objet->fichier;
    $dossier=preg_replace('#/$#','',$dossier).'/';
    $ini_objet->m_fichier($dossier,$_GET['fichier']);
    header('Location:./ini.php5');
}
if(isset($_POST['clear']))
{
    $ini_objet->clear();
}

```

```

    }
    if(isset($_POST['fichier']))
    {
        if($_POST['fichier']=='Changer')
        {
            $ini_objet->m_fichier(stripslashes($_POST['source']));
        }
        elseif($_POST['fichier']=='Supprimer')
        {
            $ini_objet->s_fichier();
        }
    }
    if(isset($_POST['groupe']))
    {
        if($_POST['groupe']=='Changer')
        {
            $ini_objet->m_groupe(stripslashes($_POST['source']));
        }
        elseif($_POST['groupe']=='Supprimer')
        {
            $ini_objet->s_groupe();
            $ini_objet->save();
        }
    }
    if(isset($_POST['item']))
    {
        if($_POST['item']=='Changer')
        {
            $ini_objet->m_item(stripslashes($_POST['source']));
        }
        elseif($_POST['item']=='Supprimer')
        {
            $ini_objet->s_item();
            $ini_objet->save();
        }
    }
    if(isset($_POST['valeur']))
    {
        if($_POST['valeur']=='Changer')
        {
            $ini_objet->m_put(stripslashes($_POST['source']));
            $ini_objet->save();
        }
        elseif($_POST['valeur']=='Supprimer')
        {
            $ini_objet->m_put('');
            $ini_objet->save();
        }
    }

    $_SESSION['ini']=serialize($ini_objet);
?>

```

Pas plus compliqué : pour chaque bouton du formulaire, la fonction (méthode de notre objet ini) correspondante est appelée et on sauvegarde la modification avec save().

Conclusion

Ce chapitre se passe de Q.C.M. Familiarisez-vous avec l'interface d'administration et découvrez dans les exemples (chapitre suivant) toutes les possibilités du .ini.

Vous êtes maintenant armés d'une page d'administration toute neuve.

Exemples

Voici quelques exemples d'utilisation du `.ini` sur votre site. On suppose dans chaque cas que vous avez inséré la class `ini` sur votre page.

Enregistrer des notes

Voici de quoi faire un système de notation de cours, de téléchargement, de vidéo, de récit... J'ai choisi *vidéo* pour l'exemple, mais vous pourrez facilement l'adapter à autre chose.

Code à placer dans la balise head

Le script suivant permettra de colorer les étoiles quand le pointeur passe au dessus :

Code : JavaScript

```
<script type="text/javascript">
<!--
var etoile=new Image();
etoile.src="site_image/etoile.gif";
function ov(url,id)
{
    for(i=1;i<=id;i++)
    {
        document.getElementById(url+i).src=etoile.src;
    }
    document.getElementById('count'+url).innerHTML=nombre;
    return true;
}
function out(url)
{
    document.getElementById('count'+url).innerHTML='0';
    for(i=1;i<=5;i++)
    {
        document.getElementById(id+i).src="site_image/etoileb.gif";
    }
    return true;
}
function vote(url, id)
{
    window.location.href="?page=4111&vote="+url+"&note="+id;
    return true;
}
-->
</script>
```

Code au niveau de la vidéo PHP + HTML

Code : PHP

```
<?php
/* Code pour chaque vidéo */
$url="video/super_drole.wmv";
echo '<embed type="application/x-mplayer2" width="512px"
height="384px" autostart="1" showcontrols="1" src="'. $url. '" /><br
/>
Note des visiteurs : ' ;
$note=array(); $total=array();
$note[$lettre]=0;
$total[$lettre]=0;
for($i=1;$i<=5;$i++)
{
    $valeur=$ini_ob->m_valeur($url,$i);
    $note[$lettre]+=( $i*$valeur);
    $total[$lettre]+=$valeur;
}
```

```

if($total[$lettre]==0)
{
    echo 'Aucune note';
}
else
{
    // On arrondit la note sur 10 (en 10 demi-étoiles)
    $dix=round($note[$lettre]*2/$total[$lettre]);
    for($k=1;$k<=5;$k++)
    {
        if($k*2<=$dix)
            echo '';
        elseif($k*2-1<=$dix)
            echo '';
        else
            echo '';
    }
    echo ' ' . ($dix/2);
    // Les images sont en dessous si vous ne voulez pas faire
    les vôtres
}
?>

```

- Les étoiles : ★★☆☆

Code : HTML

```

Votre note : <span onmouseout="javascript:out('<? echo $url; ?>');"
style="cursor:pointer;"><img onclick="javascript:vote('<? echo $url;
?>',1);" id="<? echo $url; ?>1" src="site_image/etoileb.gif"
onmouseover="javascript:ov('<? echo $url; ?>',1);" /><img
onclick="javascript:vote('<? echo $url; ?>',2);" id="<? echo $url; ?
>2" src="site_image/etoileb.gif" onmouseover="javascript:ov('<? echo
$url; ?>',2);" /><img onclick="javascript:vote('<? echo $url; ?
>',3);" id="<? echo $url; ?>3" src="site_image/etoileb.gif"
onmouseover="javascript:ov('<? echo $url; ?>',3);" /><img
onclick="javascript:vote('<? echo $url; ?>',4);" id="<? echo $url; ?
>4" src="site_image/etoileb.gif" onmouseover="javascript:ov('<? echo
$url; ?>',4);" /><img onclick="javascript:vote('<? echo $url; ?
>',5);" id="<? echo $url; ?>5" src="site_image/etoileb.gif"
onmouseover="javascript:ov('<? echo $url; ?>',5);" /></span> <span
id="count<? echo $url; ?>">0</span>

```

Code à mettre tout au début de la page

Code : PHP

```

<?php
require "require/class.ini.php";
$ini_ob=new ini();
$ini_ob->m_fichier('site_image/cours.ini');
if(!isset($_SESSION['identifiant']))
$_SESSION['identifiant']=rand(111111,99999);
if(isset($_GET['vote']))
{
    $note=round(min(5,max(1,$_GET['note']))); // Dans une
variable get, il peut y avoir n'importe quoi ; donc on arrondit
(bloquer les nombres décimaux) et on limite entre 1 et 5. Donc, si
$note est différent de $_GET['note'], c'est qu'il y avait une
valeur non comptant dans $_GET['note'].
    if($note==$_GET['note'] && false===$ini_ob-
>m_valeur($_SESSION['identifiant'],$_GET['vote']))
    {

```

```

        $ini_ob->m_groupe($_GET['vote']);
        $ini_ob->m_item($note);
        $ini_ob->m_put($ini_ob->valeur+1);
        $ini_ob->m_groupe($_SESSION['identifiant']);
        $ini_ob->m_item($_GET['vote']);
        $ini_ob->m_put($note);
        $ini_ob->save();
        echo 'Note enregistrée : '.$note.'  
';
    }
}
?>

```

Système de commentaires

Si par exemple vous avez des news que l'on peut différencier grâce à un index (numéro qui s'incrémente à chaque nouvelle news), voici comment procéder :



D'abord, rappelons-le dans le langage .ini, une ligne = une instruction ; alors comment faire rentrer un textarea multi-lignes dans une valeur item-groupe ?

Avec deux fonctions : `nl2br()` et `preg_replace()`.

Code : PHP

```

<?php
$com=nl2br($_POST['commentaire']); /* On met une balise <br /> à
chaque nouvelle ligne */
$com=preg_replace('#\n#','',$com); /* On supprime toutes les entrées
à la ligne */
?>

```

Code complet

Code : PHP

```

<?php
/* Ici on affiche la news */
$id_news=3; // à remplacer par l'index de la news
$obj_ini=new ini();
$obj_ini->m_fichier("commentaires.ini");
if(isset($_POST['commentaire']))
{
    $com=preg_replace('#\n#','',$com, nl2br(htmlspecialchars($_POST['commentaire'])));
    $obj_ini->m_put($com,time(),$id_news);
    $obj_ini->save();
}
// Puis on affiche les commentaires
foreach($obj_ini->array_groupe($id_news) as $time=>$commentaire)
{
    echo '<hr />'; // On met un autre signe de séparation
    echo 'Message posté le '.date('d/m/Y à H:i',$time).'<br />
    '.$commentaire;
}
?>

```

Placez le formulaire à la suite :

Code : HTML

```

<form method="post" action="">
    <label>Poster un commentaire : <br />
    <textarea rows="4" cols="60"

```

```
name="commentaire"></textarea></label>
</form>
```

Toutefois, quand vous aurez plus de données à insérer (pseudo, un index des commentaires, un titre...), je vous conseille d'utiliser la méthode suivante :

- pour insérer :

Code : PHP

```
<?php $obj_ini-
>m_put(time().'##'.$pseudo.'##'.$titre.'##'.$com,$id_com,$id_news);
?>
```

- et pour extraire :

Code : PHP

```
<?php
list($time,$pseudo,$titre,$commentaire)=explode('##',$commentaire,4);
/* Le 4 correspond au nombre de variables dans list(), il faut le
mettre sinon lorsqu'un utilisateur mettra ## dans son commentaire,
la fin du commentaire manquera.*/
?>
```

Autres idées

Voilà une petite liste non exhaustive de ce que vous pouvez encore faire avec l'objet ini :

- système de news complet,
- avatars pour les membres,
- tableau de scores pour un jeu en ligne,
- galerie d'images,
- tribune libre,
- livre d'or.

Vous aurez, j'en suis sûr, bien assez d'idées pour utiliser le nouvel outil que vous détenez maintenant !

À nouveau, pas besoin de Q.C.M pour ce chapitre. Si vous avez d'autres idées pour améliorer la class ini (d'autres méthodes à ajouter ou des variables complémentaires), postez un commentaire ! 🤔

C'était la dernière partie : j'espère qu'on se reverra sous peu dans un autre tuto. 😎

Voilà : j'espère que ce tutoriel aura pu vous être utile. 🤔

Merci pour leurs commentaires très utiles à :

- Lo-X
- Aztek
- samuel2202
- berdes l