Classe abstraite

« Log »

METHODES DE LOG
log.php

Version doc / classe :
1.16 / 1.15

METHODES STATIQUES

f ()

Crée un fichier nommé « $fileName.html » et y place le résultat de krumo(...$values)
krumo() est en résumé un var_dump() amélioré. D'autant plus que sur certains environnements, var_dump() peut afficher un résultat pas très human-readable.

```
$val = 'valeur_enorme';
 Log::f('mon_premier_log', $val);
// crée 'mon_premier_log.html' qui contient 'valeur_enorme'
```

append_f ()

Idem que f () mais conserve le contenu initial du fichier et ajoute à la fin : la date et l'heure du log ainsi que le nouveau contenu

```
Log::append_f('mon_premier_log' , 'nouveau_contenu_a_ajouter');
// ajoute 'nouveau_contenu_a_ajouter' à la fin de 'mon_premier_log.html'
// Si le fichier n'existe pas, il est créé
```

f1 ()

Idem que f () mais avec un fichier nommé 'f1.html'.

f2 ()

Idem que f1 () mais avec un fichier nommé 'f2.html'.

append_f1 ()

Idem que append_f () mais avec un fichier nommé 'append_f1.html'.

append_f2 ()

Idem que append_f1 () mais avec un fichier nommé 'append_f2.html'.

var_dump_f ()

Crée un fichier d'un nom donné et y place le résultat de var_dump(...$values)

$val = 'valeur_enorme';
Log::var_dump_f('mon_premier_log', $val);
// crée 'mon_premier_log.html' qui contient 'valeur_enorme'

var_dump_f1 ()

Crée un fichier nommé 'var_dump_f1.html' et y place le résultat de var_dump(...$values) en conservant le contenu initial du fichier

$val = 'toto';
Log::var_dump_f1($val);
// crée 'vdf1.html' qui contient 'toto'

var_dump_append_f ()

Idem que var_dump_f () mais conserve le contenu initial du fichier et ajoute à la fin la date et l'heure du log ainsi que le nouveau contenu

Log::var_dump_append_f('mon_premier_log' , 'nouveau_contenu_a_ajouter');
// ajoute 'nouveau_contenu_a_ajouter' à la fin de 'mon_premier_log.html'
// Si le fichier n'existait pas, il aurait été créé

hacking_attempt ()

A utiliser pour logger une tentative de piratage qui aurait été détectée. Le log est mémorisé dans « logs/hacking_attemps/ » et dans un dossier correspondant au mois et à l'année de la tentative de piratage. Dans un fichier '.hrml' intégrant la date et l'heure. La date et l'heure y sont enregistrés avec $_SESSION, $_POST, $_GET, debug_backtrace. Ce dernier contient également les arguments transmis à hacking_attempt().

Log:: hacking_attempt (« tentative d'insertion d'un script js »);

error ()

Cette méthode permet de faire un log dans 'php_error.log', mais vous permet de fournir plusieurs arguments. Un retour à la ligne est intégré et une mise en évidence

appliquée pour que l'erreur soit plus rapidement aperçue visuellement. La date et l'heure de l'erreur est aussi intégrée. Tous les arguments fournis sont json_encodés, ce qui vous permet de différencier les valeurs numériques des chaînes de texte.

```
$userInfo = [
        'id' => null,
        'age' => 31,
        'age_str' => '31'
];
Log:: error (« id utilisateur manquant », $userInfo);
// Cela ajoutera la date et l'heure actuelle comme habituellement, puis « id utilisateur manquant », et $userInfo dans 'php_error.log' comme ceci :
[01-Mar-2021 15:14:45 UTC]
_ERR:
"id utilisateur manquant"
{"id":null,"age":31,"age_str":"31"}
```

INSTALLATION SUR LE FRAMEWORK développé Jean-Jacques Pagan @Jijou

1 / Editer le fichier de config .ini

Cette étape est optionnelle car si *PATH_LOGS n'est pas défini dans le fichier de config .ini, il sera auto-généré.*

Ouvrir le fichier de config .ini du projet.
Ajouter la ligne 'PATH_LOGS' sous 'PATH_FILES'.

En local, ce fichier est situé dans le dossier $wamp (ou autre outil) suivi de « /files/ »

```
...
PATH_FILES   = files/nomduprojet/HTML/
PATH_LOGS    = files/nomduprojet/logs/   ⇦ ligne à ajouter*
...
```

Si la ligne « PATH_LOGS » n'est pas définie : PATH_LOGS vaudra PATH_HOME suivi de « files », suivi de BASE_HREF suivi de « logs ».

Dans l'exemple ci-dessus, les logs seront enregistrés dans « $dossier_du_projet/files/logs/ »

2 / Profiter

Sur certains projets comme afpacar, la classe est déjà intégrée au framework depuis route.php : vous pouvez l'utiliser dès maintenant.

3 / Précisions

Penser à ajouter le dossier « logs » au « .gitignore » pour que les fichiers de log ne soient pas pushés lors des repositories git.

Seuls les logs effectués en localhost seront effectués. En production, les logs ne se feront pas, sauf si l'utilisateur connecté est un développeur du projet.

Les logs ont pour but principal d'aider au développement, bien penser supprimer les fichiers « logs » (et la ligne de log) dès que vous n'en n'avez plus besoin.

Des fichiers trop nombreux ou un fichier trop imposant (particulièrement dans le cas d'appel à des méthodes utilisant « append ») ralentissent le serveur et peuvent le faire planter.

Made in France
with ♥

#happy_coders

Auteur : Damien Grember