Le filtrage et l'échappement des données en PHP 5 : documentation de la technique du Poka-Yoké

par Guillaume Rossolini (Tutoriels Web)

Date de publication : 5 juin 2007

Dernière mise à jour :

Ce framework est destiné à protéger vos applications PHP 5 : il **filtre** les données saisies par l'utilisateur et se charge de **préparer les valeurs** (conversion, échappement) en vue d'un affichage dans vos documents ou d'une utilisation dans des fonctions.
Vous trouverez un tutoriel d'initiation dans mes autres articles : **http://g-rossolini.developpez.com/tutoriels/php/poka-yoke/**

Je n'ai pas finalisé ce framework. Les tests n'ont pas tous été faits, je n'ai pas tout validé. Je publie cette documentation et mes sources simplement parce que je ne donnerai plus suite au projet mais je n'aimerais pas pour autant avoir perdu tout ce travail. Si cela peut en inspirer certains, ce sera parfait. Pour ma part, j'arrête de développer un framework à part et je m'aligne sur les frameworks existants.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

I - Introduction

I-A - Remerciements

Je remercie chaleureusement tous ceux qui m'ont aidé lors de la conception de ce framework : **pcaboche**, **Yoshio**, **wamania** et **doctorrock**, sans oublier Marco Tabini pour avoir fourni l'idée de départ.

I-B - Analyse

Diagramme des classes - UML

Le diagramme ci dessus devrait contenir bien plus d'éléments mais j'estime que cela compliquerait trop la lecture. Il suffira de comprendre que la classe **DVP_FilterException** est dérivée en autant de classes que nécessaires, et que ce sont ces classes dérivées qui sont utilisées dans les classes du framework. Les propriétés et méthodes sont détaillées dans la suite de cette documentation, ainsi que la liste des exceptions déclenchées par chaque méthode.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

II - DVP_Iterator

Cette classe est nécessaire à l'implémentation d'**IteratorAggregate** par **DVP_DataFilter**. Il n'y a pas de raison de l'utiliser explicitement en dehors de cela.

II-A - Implémentations

Iterator

Cette interface est utilisée par l'interface **IteratorAggregate** implémentée par la classe **DVP_DataFilter**.

Liste des méthodes nécessaires :

- **rewind()** : Le pointeur de fichier revient en arrière d'une position ;
- **current()** : Retourne le pointeur d'Array ;
- **key()** : Retourne l'index du pointeur d'Array ;
- **next()** : Le pointeur d'Array avance d'une position ;
- **valid()** : Vérifie la validité du pointeur d'Array.

II-B - Propriétés "private"

Liste des propriétés :

- **data** : Un Array des données à parcourir.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

III - DVP_DataFilter

*Toutes les valeurs par défaut sont définies dans les constantes de la classe **DVP_DataFilter**.*

III-A - Implémentations

ArrayAccess

L'interface **ArrayAccess** permet au développeur d'utiliser un objet PHP 5 comme s'il était de type Array.

Liste des méthodes nécessaires :

- **offsetExists($field)** : Vérifie si un champ existe dans l'objet ;
- **offsetGet($field)** : Lance une exception car nous voulons obliger le développeur à utiliser les méthodes "get..." ;
- **offsetSet($field, $value)** : Assigne une valeur à un champ de l'objet ;
- **offsetUnset($field)** : Supprime un champ de l'objet.

Countable

Cette interface permet de définir exactement quelles propriétés de l'objet sont utilisées lorsque la fonction **count()** est utilisée.

Liste des méthodes nécessaires :

- **count()** : Retourne le nombre de champs qui ont été filtrés.

IteratorAggregate

Cette interface permet d'utiliser **foreach()** sur l'objet.

Liste des méthodes nécessaires :

- **getIterator()** : Retourne un objet **DVP_Iterator**.

III-B - Propriétés "protected"

D'intérêt général :

- **data** : L'Array de données ;
- **filteredData** : Un Array indiquant les champs filtrés.

Groupes de valeurs autorisées :

- **allowedCharsets** : Un Array des jeux de caractères autorisés ;
- **allowedDBMS** : Un Array des SGBD autorisés ;
- **allowedPCREModes** : Un Array des modes de filtrage PCRE.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Valeurs par défaut :

- **defaultCharset** : Le jeu de caractères par défaut ;
- **defaultDBMS** : Le SGBD par défaut.

III-C - Constantes

Valeurs par défaut :

- **DEFAULT_ARRAY** = array()
- **DEFAULT_CHARSET** = **DVP_DataFilter::CHARSET_LATIN_1**
- **DEFAULT_DBMS** = **DVP_DataFilter::DBMS_MYSQL**
- **DEFAULT_HEX** = 0x0
- **DEFAULT_INT** = 0
- **DEFAULT_NUMERIC** = 0
- **DEFAULT_STRING** = ''

Utilisées par htmlentities() :

- **QUOTE_STYLE** = ENT_QUOTES

Destinées à remplir filteredData :

- **FILTERED_NO_CHANGE**
- **FILTERED_MODIFIED**

Jeux de charactères :

- **CHARSET_LATIN_1** = 'iso-8859-1'
- **CHARSET_LATIN_9** = 'iso-8859-15'
- **CHARSET_UTF_8** = 'utf-8'

SGBDs :

- **DBMS_MSSQL**
- **DBMS_MYSQL**
- **DBMS_OCI**
- **DBMS_ODBC**
- **DBMS_ORA**
- **DBMS_POSTGRESQL**
- **DBMS_PDO**
- **DBMS_SQLITE**

Utilisées dans filterUsingArray() :

- **ARRAY_ARRAYS**
- **ARRAY_NUMERICS**
- **ARRAY_INTS**
- **ARRAY_STRINGS**

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Utilisées dans filterUsingPCRE() :

- **PCRE_MATCH**
- **PCRE_REPLACE**

Relatives aux expressions rationnelles :

- **REGEX_DELIMITER** = '/'
- **REGEX_EMAIL** = '#^\w+(?:[-+.]\w+)*@\w+(?:[-.]\w+)*\.\w+$#'
- **REGEX_NEWLINES** = '/[\n\r]/'
- **REGEX_URI** = '`^(([^:/?#]+):)?(//([^/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?$`'
- **REGEX_TEXT_ISO** = '/[^a-z0-9
  ?!:;.,-_()[]{}\/
  ¡¢£¤¥¦§¨©ª«¬-®¯°±²³´µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñòóôõö÷øùúûüý
  ¡'

Lorque je ne précise pas la valeur d'une constante, cela signifie qu'elle est totalement arbitraire et à usage uniquement interne.

III-D - Méthodes "protected"

checkField()

Accès : "**protected**".

C'est l'une des méthodes fondamentales de **DVP_DataFilter**. Elle vérifie qu'un champ existe bel et bien dans l'objet **DVP_DataFilter**. Elle est appelée par la majorité des autres méthodes, en particulier les testeurs, les filtres et les getteurs.

Paramètres :

- **field** : Le champ à vérifier.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

clean()

Accès : "**protected**".

Cette méthode supprime de l'objet les champs qui n'ont pas été filtrés avec succès.

Paramètres :

- *aucun*

Exceptions :

- *aucune*

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

III-E - Méthodes "public"

III-E-1 - Les testeurs

isArray()

Permet de savoir si un champ de l'objet est de type Array.

Retourne TRUE ou FALSE.

Paramètres :

- **field** : Le champ à vérifier.

Exceptions :

- *aucune*

Exemple d'utilisation :

```
if($_POST->isArray('checkboxes'))
{
    echo 'vrai';
}
else
{
    echo 'faux';
}
```

isEmpty()

Permet de savoir si un champ de l'objet est vide.

Retourne TRUE ou FALSE.

Paramètres :

- **field** : Le champ à vérifier.

Exceptions :

- *aucune*

Exemple d'utilisation :

```
if($_POST->isEmpty('login'))
{
    echo 'vrai';
}
else
{
    echo 'faux';
}
```

isInt()

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Permet de savoir si un champ de l'objet est de type numérique entier.

Retourne TRUE ou FALSE.

Paramètres :

- **field** : Le champ à vérifier.

Exceptions :

- *aucune*

Exemple d'utilisation :

```
if($_POST->isInt('id'))
{
    echo 'vrai';
}
else
{
    echo 'faux';
}
```

isNumeric()

Permet de savoir si un champ de l'objet est de type numérique quelconque.

Retourne TRUE ou FALSE.

Paramètres :

- **field** : Le champ à vérifier.

Exceptions :

- *aucune*

Exemple d'utilisation :

```
if($_POST->isNumeric('mark'))
{
    echo 'vrai';
}
else
{
    echo 'faux';
}
```

isString()

Permet de savoir si un champ de l'objet est de type String.

Retourne TRUE ou FALSE.

Paramètres :

- **field** : Le champ à vérifier.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Exceptions :

- *aucune*

Exemple d'utilisation :
```
if($_POST->isString('login'))
{
    echo 'vrai';
}
else
{
    echo 'faux';
}
```

III-E-2 - Les filtres

Ces méthodes ne peuvent être utilisées que sur des champs qui ont été filtrés au préalable.

filterArray()

S'assure que le champ est un tableau.

Paramètres :

- **field** : De type String, c'est le nom du champ à filtrer ;
- **type** : De type String, c'est le nom du champ à filtrer ;
- **replacement** (par défaut = **DVP_DataFilter::DEFAULT_ARRAY**) : De type Array, c'est la valeur retournée si le champ n'est pas de type Array.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException
- DVP_ValueException

Exemple d'utilisation :
```
$_POST->filterArray('checkboxes', DVP_DataFilter::ARRAY_INTS);
```

filterEmail()

S'assure que le champ est de type String et qu'il contient une adresse e-mail (syntaxiquement valide).

Dépendance : **DVP_DataFilter::filterUsingPCRE()**.

Paramètres :

- **field** : De type String, c'est le nom du champ à filtrer ;
- **replacement** (par défaut = **DVP_DataFilter::DEFAULT_STRING**) : De type String, c'est la valeur retournée si le champ n'est pas de type String ou ne correspond pas à un e-mail.

Exceptions :

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

- DVP_AbsentOffsetException
- DVP_RegexException
- DVP_TypeException
- DVP_ValueException

Exemples d'utilisation :
```
$_POST->filterEmail('email');
$_POST->filterEmail('email', 'admin@site.com');
```

filterHex()

S'assure que le champ est un valeur hexadécimale.

Paramètres :

- **field** : De type String, c'est le nom du champ à filtrer ;
- **replacement** (par défaut = **DVP_DataFilter::DEFAULT_HEX**) : De type hexadécimal, c'est la valeur retournée si le champ n'est pas de type hexadécimal.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

Exemple d'utilisation :
```
$_POST->filterHex('ip_address');
```

filterInt()

S'assure que le champ est une valeur numérique entière.

Paramètres :

- **field** : De type String, c'est le nom du champ à filtrer ;
- **replacement** (par défaut = **DVP_DataFilter::DEFAULT_INT**) : De type Integer, c'est la valeur retournée si le champ n'est pas de type Integer.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

Exemples d'utilisation :
```
$_POST->filterInt('id');
$_POST->filterInt('id', 1);
```

filterNumeric()

S'assure que le champ est une valeur numérique.

Paramètres :

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

- **field** : De type String, c'est le nom du champ à filtrer ;
- **replacement** (par défaut = **DVP_DataFilter::DEFAULT_NUMERIC**) : De type numérique quelconque, c'est la valeur retournée si le champ n'est pas de type numérique quelconque.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

Exemples d'utilisation :

```
$_POST->filterNumeric('mark');
$_POST->filterNumeric('mark', 1);
```

filterSingleLine()

S'assure que le champ est de type String contenant une seule ligne de texte.

Dépendance : **DVP_DataFilter::filterUsingPCRE()**.

Paramètres :

- **field** : De type String, c'est le nom du champ à filtrer ;
- **replacement** (par défaut = **DVP_DataFilter::DEFAULT_STRING**) : De type String, c'est la valeur retournée si le champ n'est pas de type String ou s'il contient plusieurs lignes.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

Exemples d'utilisation :

```
$_POST->filterSingleLine('login');
$_POST->filterSingleLine('login', 'guest');
```

filterURI()

S'assure que le champ est de type String et qu'il correspond à une URI.

Dépendance : **DVP_DataFilter::filterUsingPCRE()**.

Paramètres :

- **field** : De type String, c'est le nom du champ à filtrer ;
- **replacement** (par défaut = **DVP_DataFilter::DEFAULT_STRING**) : De type String, c'est la valeur retournée si le champ n'est pas de type String ou s'il contient plusieurs lignes.

Exceptions :

- DVP_AbsentOffsetException
- DVP_RegexException
- DVP_TypeException
- DVP_ValueException

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Exemples d'utilisation :
```
$_POST->filterURI('website');
$_POST->filterURI('website', 'http://g-rossolini.developpez.com/');
```

filterUsingArray()

S'assure que le champ est du type $type et présente dans le tableau $valeursBlanches.

Paramètres :

- **field** : De type String, c'est le nom du champ à filtrer ;
- **whiteValues** : De type Array, il contient les seules valeurs que le champ sera autorisé à prendre ;
- **type** : C'est l'une des constantes **DVP_DataFilter::ARRAY_***, ce paramètre contraint le champ à un type défini.

Exceptions :

- DVP_AbsentOffsetException
- DVP_ArrayTypeException
- DVP_TypeException
- DVP_ValueException

Exemples d'utilisation :
```
$_POST->filterUsingArray('id', array(1, 3, 7), DVP_DataFilter::ARRAY_INTS); // L'ID est soit 1, soit 3, soit 7
$_POST->filterUsingArray('login', array('Yogui', 'BrYs'), DVP_DataFilter::ARRAY_STRINGS); // Le login est soit "Yogui", soit "BrYs"
```

Vous ne devez pas fournir de tableau de valeurs blanches contenant plusieurs types. Un champ spécifique doit être d'un type bien défini. Il ne peut pas être "d'un type ou d'un autre" car cela donnerait lieu à des failles de sécurité.

filterUsingPCRE()

Vérifie que le champ vérifie une expression rationnelle.

Paramètres :

- **field** : De type String, c'est le nom du champ à filtrer ;
- **regex** : De type String, c'est l'expression rationnelle à utiliser ;
- **replacement** (par défaut = **DVP_DataFilter::DEFAULT_STRING**) : De type String, c'est la valeur retournée si le champ n'est pas de type String ou s'il ne vérifie pas l'expression rationnelle ;
- **mode** (par défaut = **DVP_DataFilter::PCRE_REPLACE**) : C'est l'une des constantes **DVP_DataFilter::PCRE_***, ce paramètre définit le comportement de la méthode filterUsingPCRE.

Exceptions :

- DVP_AbsentOffsetException
- DVP_RegexException
- DVP_TypeException
- DVP_ValueException

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Cette méthode est pratique mais son utilisation est complexe. Pour en simplifier l'utilisation, il est possible de définir d'autres methodes comme celles qui l'utilisent déjà (cf. ci-dessus).

untaint()

Marque un champ comme filtré, sans lui apporter aucune modification.

Paramètres :

- **field** : De type String, c'est le nom du champ à marquer.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

Cette méthode est nécessaire pour un nombre très faible de cas particuliers, comme par exemple les messages utilisateurs. Il faut y prendre garde, car cette méthode introduit des données non filtrées dans l'application.

III-E-3 - Les getteurs

getArray()

Retourne sous forme de type Array.

Paramètres :

- **field** : Le champ à retourner.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TaintedException
- DVP_TypeException

Exemple d'utilisation :

```
$_POST->filterArray('checkboxes', DVP_DataFilter::ARRAY_INTS);
foreach($_POST->getArray('checkboxes') as $checkbox)
{
    echo $checkbox.'<br />'; // Afficher cette valeur est sans danger car elle a déjà été filtrée
 comme "numérique entier"
}
```

getFields()

Retourne un Array contenant les noms des champs filtrés. Cette méthode est définie pour permettre l'utilisation des objets **DVP_DataFilter** dans la structure **foreach**. L'implémentation de l'interface **IteratorAggregate** rend l'appel de cette méthode transparent dans les situations comme **foreach**.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.
http://g-rossolini.developpez.com/ateliers/php/poka-yoke/doc/

Paramètres :

- **field** : Le champ à retourner.

Exceptions :

- *aucune*

Exemple d'utilisation (méthode 1) :
```
$_POST->filterArray('checkboxes', DVP_DataFilter::ARRAY_INTS);
$_POST->filterSingleLine('login');

foreach($_POST->getFields() as $field)
{
    echo $field.'<br />';
}
```

Exemple d'utilisation (méthode 1) :
```
$_POST->filterArray('checkboxes', DVP_DataFilter::ARRAY_INTS);
$_POST->filterSingleLine('login');

foreach($_POST as $field)
{
    echo $field.'<br />';
}
```

*Le retour de la méthode ne contient aucune "valeur", uniquement les noms des champs ; ou plutôt, les valeurs **sont** les noms des champs, ce qui rend leur index inutile :*

L'index de chaque champ est inutile :
```
$_POST->filterArray('checkboxes', DVP_DataFilter::ARRAY_INTS);
$_POST->filterSingleLine('login');

foreach($_POST as $index => $field)
{
    echo $index.' '.$field.'<br />';
}
```

getHTML()

Retourne le champ sous forme d'entités HTML. Le format des entités est défini au niveau de l'objet **DVP_DataFilter**.

Paramètres :

- **field** : Le champ à retourner.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException
- DVP_ValueException

Exemple d'utilisation :
```
$_POST->filterSingleLine('login');
echo $_POST->getHTML('login');
```


Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

getRaw()

Retourne un champ sous sa forme originale.

Paramètres :

- **field** : Le champ à retourner.

Exceptions :

- DVP_AbsentOffsetException
- DVP_ArrayAccessException
- DVP_TaintedException
- DVP_TypeException

getRegex()

Retourne un champ sous forme de chaîne prête à utiliser dans une expression rationnelle.

Paramètres :

- **field** : Le champ à retourner ;
- **delimiter** (par défaut = **DVP_DataFilter::REGEX_DELIMITER**) : Le délimiteur de la regex.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

Exemple d'utilisation :

```
$_POST->filterSingleLine('login');
$regex = '#<user>'.$_POST->getRegex('login', '#').'.*</user>#Ui'; // Exemple :
 #<user>Yog.*</user>#Ui

$users_xml = file_get_contents('users.xml');

if(preg_match($users_xml, $regex))
{
    echo 'vrai';
}
else
{
    echo 'faux';
}
```

users.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
    <user>Yogui</user>
    <user>BrYs</user>
</users>
```

L'exemple ci-dessus vérifie si un nom d'utilisateur (soumis par formulaire) existe dans un document XML contenant une liste d'utilisateurs.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

getShellArgument()

Retourne un champ sous forme de chaîne prête à utiliser comme argument d'une ligne de commande système.

Paramètres :

- **field** : Le champ à retourner.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

getShellCommand()

Retourne un champ sous forme de chaîne prête à utiliser comme commande système.

Paramètres :

- **field** : Le champ à retourner.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

getSQL()

Retourne un champ sous forme de chaîne prête à utiliser dans une requête (SQL) en fonction du SGBD utilisé.

Paramètres :

- **field** : Le champ à retourner ;
- **dbms** (par défaut = **DVP_DataFilter::DEFAULT_DBMS**) : Le SGBD de destination ;
- **argument** (par défaut = NULL) : Pour PostgreSQL, il prend comme valeur "string" ou "bytea" ; pour MySQL, il correspond à l'identifieur de ressource.

Exceptions :

- DVP_DBMSEscapeException
- DVP_AbsentOffsetException
- DVP_PostgreSQLException
- DVP_TypeException
- DVP_ValueException

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

IV - DVP_HTTPUploadFilter

DVP_HTTPUploadFilter hérite de **DVP_DataFilter**.

*L'extension PHP **fileinfo** est nécessaire au bon fontionnement de cette classe (gestion fiable des types MIME).*

IV-A - Propriétés "protected"

Intérêt général :

- **destination** : Le chemin système de destination ;
- **finfo** : L'objet Finfo utilisé pour contrôler le type MIME des fichiers ;
- **refusedData** : Un Array des fichiers refusés par **DVP_HTTPUploadFilter**.

Groupes de valeurs autorisées :

- **allowedMimes** : PDF + MP3 + TEXT ;
- **allowedMimeGroups** : ARCHIVES + DOCUMENTS + IMAGES + SCRIPTS ;
- **allAllowedMimes** : allowedMimes + allowedMimeGroups.

Format de nombres :

- **decimalsSeparator** : Séparateur de décimales (en Français, il s'agit de la virgule ; en Anglais, c'est le point) ;
- **thousandsSeparator** : Séparateur de milliers (en Français, il s'agit de l'espace ; en Anglais, c'est la virgule).

Les deux dernières variables présentées ici sont déclarées "final". Elles ne peuvent pas etre modifiées en-dehors du constructeur de la classe.

IV-B - Constantes

Résultat de l'opération de filtrage :

- **FILTERED_ACCEPTED**
- **FILTERED_REFUSED**

Configuration de l'extension fileinfo :

- **MAGIC_FILE** : Le chemin d'accès au ficher "magique" ;
- **FILEINFO_MODE** : Le mode d'utilisation du fichier "magique" (constantes du scope global : FILEINFO_MIME ou FILEINFO_NONE).

Types MIME suportés :

- **MIME_MP3**
- **MIME_PDF**
- **MIME_TEXT**
- **MIME_ARCHIVES**
- **MIME_DOCUMENTS**
- **MIME_IMAGES**

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

- **MIME_SCRIPTS**

Tailles des fichiers par type MIME :

- **SIZE_MP3**
- **SIZE_PDF**
- **SIZE_TEXT**
- **SIZE_ARCHIVES**
- **SIZE_DOCUMENTS**
- **SIZE_IMAGES**
- **SIZE_SCRIPTS**

Valeurs par défaut :

- **DEFAULT_SEPARATOR_DECIMALS** = ','
- **DEFAULT_SEPARATOR_THOUSANDS** = ' '

Unités pour le formatage des tailles de fichiers :

- **UNIT_OCTET**
- **UNIT_KILO_OCTET**
- **UNIT_MEGA_OCTET**
- **UNIT_GIGA_OCTET**

IV-C - Méthodes "private"

cleanString()

Cette méthode n'est utilisée qu'en interne. Il n'y a pas de raison de permettre au développeur de l'utiliser.

Cette méthode est finale, elle ne peut être surchargée (même si on la définit en *protected* ou *public*).

Paramètres :

- **string** : Le nom de fichier à formater.

Exceptions :

- *aucune*

IV-D - Méthodes "protected"

filterFile()

Cette méthode permet de vérifier la validité d'un fichier en fonction de son type et de sa taille. Si les vérifications sont couronnées de succès, le fichier est automatiquement renommé et déplacé hors du répertoire temporaire du serveur Web.

Paramètres :

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

- **field** : Le champ décrivant le fichier à vérifier ;
- **intendedMime** : Le type du fichier ;
- **intendedMaxSize** : La taille maximum du fichier.

Exceptions :

- DVP_AbsentOffsetException
- DVP_FileSizeException
- DVP_TypeException
- DVP_ValueException

inArray()

Cette méthode sert à déterminer si une chaîne est contenue dans n'importe quelle chaîne d'un tableau.

Paramètres :

- **needle** : L'aiguille ;
- **haystack** : La meule de foin.

Exceptions :

- DVP_TypeException

IV-E - Méthodes "public"

IV-E-1 - Les filtres

filterArchive()

Cette méthode appelle **filterFile()** avec des paramètres prédéfinis pour les archives compressées.

Paramètres :

- **field** : Le champ décrivant le fichier à vérifier ;
- **size** (par défaut = **DVP_HTTPUploadFilter::SIZE_ARCHIVES**).

Exceptions :

- DVP_AbsentOffsetException
- DVP_FileSizeException
- DVP_TypeException
- DVP_ValueException

Exemple d'utilisation :

```
$_FILES->filterArchive('documents'); // "documents" correspond à un champ du formulaire destiné à
une archive compressée
```


Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

filterDocument()

Cette méthode appelle **filterFile()** avec des paramètres prédéfinis pour les documents texte.

Paramètres :

- **field** : Le champ décrivant le fichier à vérifier ;
- **size** (par défaut = **DVP_HTTPUploadFilter::SIZE_DOCUMENTS**).

Exceptions :

- DVP_AbsentOffsetException
- DVP_FileSizeException
- DVP_TypeException
- DVP_ValueException

filterImage()

Cette méthode appelle **filterFile()** avec des paramètres prédéfinis pour les images.

Paramètres :

- **field** : Le champ décrivant le fichier à vérifier ;
- **size** (par défaut = **DVP_HTTPUploadFilter::SIZE_IMAGES**).

Exceptions :

- DVP_AbsentOffsetException
- DVP_FileSizeException
- DVP_TypeException
- DVP_ValueException

filterMP3()

Cette méthode appelle **filterFile()** avec des paramètres prédéfinis pour les fichiers MP3.

Paramètres :

- **field** : Le champ décrivant le fichier à vérifier ;
- **size** (par défaut = **DVP_HTTPUploadFilter::SIZE_MP3**).

Exceptions :

- DVP_AbsentOffsetException
- DVP_FileSizeException
- DVP_TypeException
- DVP_ValueException

filterPDF()

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Cette méthode appelle **filterFile()** avec des paramètres prédéfinis pour les documents PDF.

Paramètres :

- **field** : Le champ décrivant le fichier à vérifier ;
- **size** (par défaut = **DVP_HTTPUploadFilter::SIZE_PDF**).

Exceptions :

- DVP_AbsentOffsetException
- DVP_FileSizeException
- DVP_TypeException
- DVP_ValueException

filterScript()

Cette méthode appelle **filterFile()** avec des paramètres prédéfinis pour les scripts.

Paramètres :

- **field** : Le champ décrivant le fichier à vérifier ;
- **size** (par défaut = **DVP_HTTPUploadFilter::SIZE_SCRIPTS**).

Exceptions :

- DVP_AbsentOffsetException
- DVP_FileSizeException
- DVP_TypeException
- DVP_ValueException

filterText()

Cette méthode appelle **filterFile()** avec des paramètres prédéfinis pour les fichiers de texte brut.

Paramètres :

- **field** : Le champ décrivant le fichier à vérifier ;
- **size** (par défaut = **DVP_HTTPUploadFilter::SIZE_TEXT**).

Exceptions :

- DVP_AbsentOffsetException
- DVP_FileSizeException
- DVP_TypeException
- DVP_ValueException

filterAnyType()

Cette méthode appelle **filterFile()** avec des paramètres prédéfinis pour le type MIME du fichier concerné.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Paramètres :

- **field** : Le champ décrivant le fichier à vérifier ;
- **size** (par défaut = **NULL**).

Exceptions :

- DVP_AbsentOffsetException
- DVP_FileSizeException
- DVP_TypeException
- DVP_ValueException

IV-E-2 - Les getteurs

Les méthodes "get..." ne fonctionnent qu'avec les fichiers filtrés.

getFormattedSize()

Retourne une taille formattée de manière lisible.

Paramètres :

- **field** Le champ décrivant le fichier.

Exceptions :

- *aucune*

getMime()

Retourne le type MIME d'un fichier.

Paramètres :

- **field** Le champ décrivant le fichier.

Exceptions :

- DVP_AbsentOffsetException
- DVP_TypeException

getName()

Retourne le nom d'un fichier.

Paramètres :

- **field** Le champ décrivant le fichier.

Exceptions :

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

- DVP_AbsentOffsetException
- DVP_TypeException

getRawSize()

Retourne la taille d'un fichier (en octets).

Paramètres :

- **field** Le champ décrivant le fichier.

Exceptions :

- *aucune*

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

V - Conclusion

Liens :

- Tutoriel : **Introduction au framework du Poka-Yoké (PHP 5)**, par Guillaume Rossolini ;
- Tutoriel : **Les exceptions et PHP 5**, par Guillaume Affringue ;
- Tutoriel : **Les nouveautés de PHP 5**, par Stéphane Eyskens.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 2006 - Guillaume Rossolini. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.