Exemple 2 : La valeur du champ txtComment suivant.

```
<input type="text" name="txtComment" value="<script> while (true)
alert('Erreur'); </script>">
```

sera interprétée comme une séquence de script et provoquera indéfiniment l'affichage d'une boîte message. Le navigateur lui-même devra être arrêté.

Pour éviter ces effets néfastes, ces caractères réservés doivent être traduits en symboles nommés HTML (aussi appelés entités HTML). Ainsi, le caractère < doit être transformé en <, > en > , etc.

La fonction PHP prédéfinie *htmlspecialchars* prend en charge ce traitement.

```
string  htmlspecialchars ( string   string , int   quote_style ,
string   charset )
```

Les remplacements effectués sont :

- " & " (et commercial) devient " `&` "
- " " " (guillemets doubles) devient " `&quot;` " lorsque `ENT_NOQUOTES` n'est pas utilisé.
- " ' " (single quote) devient " `&#039;` " uniquement lorsque `ENT_QUOTES` est utilisé.
- " < " (supérieur à) devient " `<` "
- " > " (inférieur à) devient " `>` "

Les guillemets simples et doubles ne sont pas systématiquement traduits : cela dépend de la valeur du paramètre optionnel quote_style.

`ENT_COMPAT` , la constante par défaut, va convertir les guillemets doubles, et ignorer les guillemets simples; `ENT_QUOTES` va convertir les guillemets doubles et les guillemets simples;

`ENT_NOQUOTES` va ignorer les guillemets doubles et les guillemets simples.

Exemple :

```
<?php
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new;
// <a href=&#039;test&#039;>Test</a>
?>
```

Fabrication d'une page HTML sécurisée

Description :

La fonction *htmlspecialchars* doit être appelée sur toute valeur en provenance de la base de données avant d'être affichée dans une page HTML. On l'appellera avec pour second argument la constante ENT_QUOTES pour convertir à la fois les guillemets doubles et simples.

Configuration du fichier php.ini

Description :

Afin de favoriser la détection des erreurs, ainsi que la portabilité d'une configuration PHP à une autre, il est obligatoire de réaliser le développement d'une application Web avec certaines directives de configuration PHP affectées aux valeurs recommandées suivantes.

Nom de la directive	Description	Valeur imposée
short_open_tag	Définit si les balises courtes d'ouverture de PHP (<? ?>) sont autorisées ou non.	Off
output_buffering	Définit l'activation ou non de la bufferisation de sortie. L'activation de la buffferisation peut être autorisée moyennant d'être dûment justifiée.	Off
error_reporting	Fixe le niveau d'erreur. Ce paramètre est un entier, représentant un champ de bits. Cette directive est renseignée en utilisant les types d'erreurs définis sous forme de constantes et en utilisant les opérateurs bits à bits & (ET), | (OU), ~ (SAUF), de même que l'opérateur booléen ! (SAUF). Le rapport d'erreur de niveau **E_NOTICE** (inclus dans **E_ALL**) durant le développement a des avantages. En terme de déboguage, les message d'alertes signalent des bogues potentiels dans le code. Par exemple, l'utilisation de valeurs non initialisées est signalée. Comme **E_STRICT**, nouveau niveau d'erreur introduit en PHP5, n'est pas inclus sans **E_ALL**, il faut explicitement l'ajouter. Il permet d'être alerté de l'utilisation de fonctions non recommandées.	E_ALL | E_STRICT

Compléments :

Les directives de configuration PHP peuvent être appliquées à 3 niveaux :

- à l'ensemble des applications Web d'un serveur Web par le biais du fichier php.ini,
- à l'ensemble des scripts PHP d'un répertoire par le biais d'un fichier caché situé dans ce répertoire et analysé par le serveur Web (directive php_flag du fichier .htaccess analysé par Apache),
- à un seul script PHP par le biais de l'appel de la fonction ini_set.

Intérêts :

Portabilité : indépendance du logiciel par rapport à son environnement

Fiabilité : robustesse

Annexe 1 – Eléments sur l'outil PHPDocumentor[4]

PHP Documentor est un outil de génération automatique de documentation à partir des commentaires inclus dans les programmes PHP.

PHPDocumentor eut être utilisé soit via la ligne de commande, soit via une interface web.

Le téléchargement de l'archive de l'outil PHP Documentor se fait là : http://sourceforge.net/projects/phpdocu/files/

Pour installer phpDocumentor, il faut décompresser l'archive dans un répertoire en respectant la structure interne des dossiers. L'utilisation de l'interface web implique de décompresser l'archive dans un dossier accessible par le serveur Web. Dans la suite du document, nous considérons que ce dossier se nomme *phpdoc*.

Les blocs de commentaires

Format d'un bloc de commentaires

La documentation exploitée par *PHP*Documentor doit se trouver dans un bloc de commentaires respectant le format suivant :

```
/**
 * Mes explications ...
 */
```

Ce bloc de commentaires est un bloc de commentaires étendu qui commence par un "/**" et présente un "*" au début de chaque ligne. Les blocs de commentaires précèdent les éléments qu'ils documentent.

Toute ligne dans un bloc de commentaires qui ne commence pas par un « * » sera ignorée.

Contenu d'un bloc de commentaires

Un bloc de commentaires contient trois segments de base dans l'ordre suivant :

- une **description courte** : débute sur la première ligne, et peut se terminer par un point ou une ligne blanche.
- une **description longue** : peut occuper autant de lignes que nécessaire.
- des **marqueurs** : des mots préfixés par le caractère @. Ils informent phpDocumentor sur la façon d'afficher la documentation. Tous les marqueurs sont optionnels, mais ils doivent respecter une syntaxe spécifique pour être interprétés correctement.

Voici quelques marqueurs à utiliser :

- **@author** : nom de l'auteur
- **@param** : type, nom et description d'un paramètre
- **@return** : type et description du résultat retourné par une fonction
- **@see** : nom d'un autre élément documenté, produisant un lien vers celui-ci
- **@link :** url
- **@todo** : changements à faire dans le futur

Eléments de code à documenter

Plusieurs éléments de code peuvent être documentés : des fichiers, des fonctions, des classes, des méthodes, des propriétés, des variables globales, des constantes.

Nous présentons ci-après un exemple concernant les fonctions.

[4] Référence site officiel : www.phpdoc.org

Une fonction est caractérisée par :

- son nom
- son type et sa valeur de retour (@return)
- ses paramètres (@param)
- sa description

Il est donc possible de la documenter ainsi :

```
/**
 * Echappe les caractères spéciaux d'une chaîne.

 * Envoie la chaîne $str échappée, càd avec les caractères considérés
 * spéciaux par MySql (tq la quote simple) précédés d'un \, ce qui annule
 * leur effet spécial

 * @param string $str chaîne à échapper
 * @return string
 */
function filtrerChainePourBD($str) {
  if ( ! get_magic_quotes_gpc() ) {
      $str = mysql_real_escape_string($str);
  }
  return $str;
}
```

filtrerChainePourBD[line 60] - **Echappe les caractères spéciaux d'une chaîne.**

```
string filtrerChainePourBD( string $str)
```

Envoie la chaîne $str échappée, càd avec les caractères considérés spéciaux par MySql (tq la quote simple) précédés d'un \, ce qui annule leur effet spécial

Tags:

return: chaîne échappée

Parameters

string **$str** chaîne à échapper
[Top]

On peut ajouter autant de lignes @param qu'il y a de paramètres.

En utilisant le template HTML:Smarty:PHP, on obtient l'extrait suivant dans la page HTML produite par l'outil phpDocumentor :

Génération de la documentation

La génération portera d'une part, sur les fichiers source à interpréter et d'autre part, sur le choix des formats de sortie.

Génération par l'interface web

Via un navigateur, *il faut accéder* au sous-dossier `/dobuilder` du dossier `/phpdoc` accessible du serveur web sur lequel il est installé. Dans le cas où navigateur et serveur web sont sur le même poste, on saisira l'url suivante : http://localhost/phpdoc/docbuilder. La page d'accueil suivante sera alors affichée :