PhpDocumentor : L'usine à documentation

2ème partie

Nous avons vu le mois dernier quels étaient les bons réflexes à prendre pour commenter ses sources en vue de générer des documentations de qualité pour des applications PHP. Dans cette seconde partie, on va s'intéresser aux fonctions plus avancées de phpDocumentor, sur la base d'exemples concrets.

Mise à jour

Premièrement, rappelons que phpDocumentor est un logiciel libre, dont le développement (très actif) est permanent. Aussi, il est préférable de mettre régulièrement à jour son installation. Pour ce faire, consultez de temps à autres l'adresse suivante : http://phpdoc.org, section " downloads ". Nous considérerons dans cet article que phpDocumentor est installé à l'adresse http://localhost/phpDocumentor. Notre version de phpDocumentor est la 1.2.2.

Sources de l'exemple

Afin de bien saisir le fonctionnement des différentes options, voici une partie des sources des deux fichiers sur lesquels nous allons travailler :

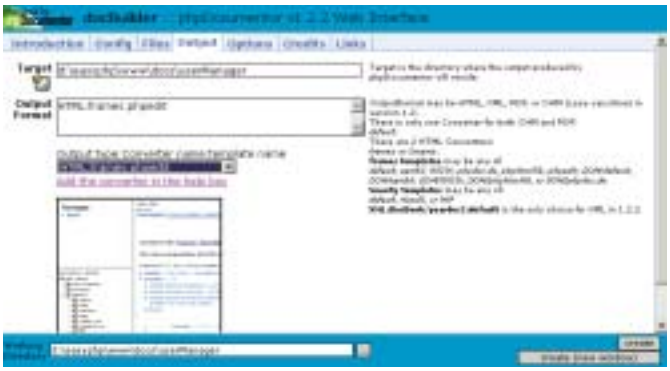
```
userManager.php :
<?
  /**
  * Pseudo gestionnaire d'utilisateurs
  *
  * Programme factice, servant à illustrer les possibilités
  * de génération de documentations par phpDocumentor
  *
  * @package userManager
  * @author Gauthier Delamarre
  */
  /**
  * Classe principale
  *
  * Cette classe va servir à manipuler les objets de type "user".
  */
  class userManager {
    /**
    * Configuration générale
    * @var array
    */
    var $_CFG;
    /**
    * Constructeur
    *
    * @param string chemin d'accès au fichier de configuration
    */
    function userManager($cfgPath) { ... }
    /**
    * Ajout d'un utilisateur
    *
    * @param object Objet de type "user"
    * @return bool
    */
    function addUser($userObj) { ... }
    /**
    * Gestion d'erreur
    *
    * @param string Message d'erreur à afficher
    * @param bool Indique si l'exécution doit êtr stoppée ou non
    */
    function raiseError($msg,$kill = TRUE) { ... }
  }
?>

user.php :
<?
  /**
  * Définition de la classe "user"
  *
  * @package userManager
  */
  /**
  * Classe définissant les utilisateurs
  */
  class user {
    /**
    * Créer un nouvel utilisateur
    *
    * @param string Nom de l'utilisateur
    * @param string Prénom
    * @param string Adresse email
    *
    * @return bool
    */
    function create($nom,$prenom,$email) { ... }
    /**
    * Charge les données utilisateurs depuis la base
    *
    * @param int Identifiant de l'utilisateur à charger
    * @return bool
    */
    function load($id) { ... }
  }
?>
```

Dans notre environnement de test, ces fichiers se situent dans d:\easy-php\www\userManager\. Nous aurons besoin de l'indiquer à phpDocumentor pour qu'il puisse les scanner. Note : Vous pourrez télécharger les fichiers complets sur programmez.com pour effectuer les tests sur votre propre machine.

Premier essai

Maintenant que nous avons installé le générateur de documentations, et que nous disposons de sources d'exemple, nous allons pouvoir créer notre première documentation. Pour cela, pointez un navigateur sur http://localhost/phpDocumentor/. L'interface web apparaît. Dans un premier temps, nous n'allons nous intéresser qu'aux onglets " files " et " output ". Dans le premier onglet, un seul paramètre est indispensable : le chemin d'accès aux fichiers. Dans le second, seul le dossier où stocker la documentation générée est obligatoire. Vous pouvez cependant en profiter pour choisir un format et/ou un thème pour votre documentation. Le choix est assez riche, et il est possible d'en sélectionner plusieurs simultanément.



indiquer le chemin d:\easyphp\www\dows\userManager pour y stocker notre documentation. Il ne nous reste plus qu'à lancer le traitement des sources pour que les documentations soient créées aux formats demandés, en cliquant sur le bouton " create ". Pour consulter le résultat, pointez un nouveau navigateur sur http://localhost/docs/userManager : Résultat de notre premier essai avec le converter HTML:frames:DOM/earthli

Note : si vous souhaitez essayer le converter CHM (format d'aide Windows), l'utilitaire Microsoft Help Workshop (gratuit, à télécharger sur le site de Microsoft).

Tuning

Nous sommes maintenant parés pour nous pencher sur les options et fonctionnalités supplémentaires de phpDocumentor. Voyons d'abord l'onglet " options " de docBuilder.

Les options ici proposées sont assez simples à comprendre, car chacune est accompagnée d'un descriptif précis. Voici un résumé de ces options :

Custom tags	Liste de tags personnalisés que l'on veut que le parser prenne en compte
Parse @access private	Détermine la prise en compte des méthodes et propriétés privées.
Generate Highlited Source Code	Ajoute à la documentation une copie colorisée des sources. De plus, chaque fonction ou classe sera liée à sa documentation, et chaque fonction PHP détectée renverra vers la documentation officielle en ligne !
JavaDoc-compliant Description Parsing	Active la compatibilité avec la syntaxe JavaDoc de descriptif des classes (la première phrase, i.e. jusqu'au premier point, correspond toujours à la description courte).
PEAR Package Repository parsing	Active la prise en compte des règles de codage spécifiques à PEAR (Chaque module est dans un répertoire propre, portant le même nom que le module, les méthodes et propriétés dont le nom débute par un _ sont considérées comme privées, et enfin, la méthode_NomDeLaClasse() est associée au destructeur de la classe.

Types personnalisés

Maintenant, voyons comment phpDocumentor gère les types de données personnalisés, à savoir, en PHP, les objets. Dans notre exemple, une fonction attend en paramètre un objet de type user. Nous avons précisé avec le tag @param que l'argument doit être un objet, mais nous n'avons précisé son type que dans la description du paramètre. Sachez que phpDocumentor accepte en fait n'importe qu'elle chaîne pour qualifier un type d'argument, et qu'il est possible de commenter ainsi :

```
@param user Objet user à ajouter au gestionnaire d'utilisateurs
```

Sous forme textuelle, cela ne change pas grand chose, certes. Mais là où cela devient intéressant, c'est que dans un cas comme celui-ci, phpDocumentor fera directement un lien depuis " user " vers la définition de la classe correspondante !

Renvois

Un autre tag essentiel est le tag @link. Celui-ci permet de générer dynamiquement des liens vers des pages spécifiques de la documentation. Vous pouvez l'utiliser dans les blocs de page comme dans les blocs de fonction, de classes ou même de variables. Exemple :

```
@param user Objet de type " user " (voir {@link user::create()} ou {@link user::load()})
```

Une autre syntaxe permet de réaliser le même type de liens, mais par le biais d'un tag à part entière, et non pas embarqué dans une description :

```
@see user::load(),user::create(),raiseError
```

Notez que la méthode raiseError() est indiquée telle quelle, sans spécifier sa classe, ni préciser qu'il s'agit d'une fonction (en ajoutant des parenthèses vides à la fin), car elle fait partie de la classe courante.

Liste de tâches

Si vous souhaitez mentionner dans vos sources les tâches qu'il reste à faire, vous pouvez ajouter des tags @todo. Ceux-ci seront ensuite regroupés en une seule page par package, pour en consulter la liste d'un coup d'oeil. Exemple :

```
@todo Ajouter le support de PostgreSQL pour l'accès à la base.
```

Fichiers README, INSTALL et CHANGELOG

Si phpDocumentor trouve un ou plusieurs de ces fichiers (sans extension), il en intégrera directement le contenu dans la documentation.

Exemples dans les commentaires

Vous avez également la possibilité de placer des exemples dans vos commentaires. Vous avez pour ce faire deux méthodes. Soit vous spécifiez le chemin d'un fichier contenant l'exemple :

```
@example exemples/exemple1.php Exemple d'utilisation
```

Soit vous placez l'exemple dans le descriptif :

```
/**
   * Charge les données utilisateurs depuis la base
   *
   * Exemple d'utilisation :
   * <code>
   * $userObj = new user;
   * $userObj->load(12);
   *</code>
...
```

Dans les deux cas, le code sera mis en couleur, et tous les termes renvoyant vers des fonctions, méthodes, variables ou constantes seront liés vers la page de documentation correspondante.

Conclusion

Nous voici arrivés au terme de notre plongée dans le monde merveilleux de phpDocumentor, qui est véritablement un outil incontournable pour qui veut documenter rapidement et efficacement ses sources. Il vous reste à essayer les différents modèles, puis à consulter la documentation officielle de phpDocumentor (www.phpdoc.org/manual) pour en tirer le meilleur parti.

■ *Gauthier Delamarre*