Manuel d'utilisation de DocFromCom (Version 1.0)

Sommaire

Conseils d'utilisation

DocFromCom se présente sous la forme d'une archive `.phar`. Plutôt que de placer cette archive à la racine de votre projet et de la copier à chaque création d'un nouveau projet, il est recommandé de placer l'archive dans le dossier où se situent tous vos projets.

Génération de documentation

Commande

Pour générer la document d'un fichier, ouvrir un terminal dans le dossier où vous avez placé l'archive `.phar` et executez la commande suivante :

```
php DocFromCom.phar document:generate "Chemin/Monfichier" "ExtensionDuFichier"
[Extension de sortie] [Langue]
```

Le chemin vers le fichier de code source peut être absolu ou relatif. Les langages disponibles pour le fichier en entrée sont :

- Tout les langages dont les commentaires multilignes sont sous la forme `/** Commentaire */` comme le c, le c++, le java, le php etc.
- Les langages dont les commentaires sont sous la forme `#Commentaire` comme le python.

Les extensions de sortie disponibles sont :

- `md` pour avoir la documentation en markdown (valeur par défault)
- `tex` pour avoir la documentation en LaTeX

Les langues disponibles pour les éléments ajoutés au fichier de sortie par DocFromCom sont :

- `EN` pour Anglais (valeur par défault)
- `FR` pour Français

Écriture des commentaires

Pour générer la documentation, DocFromCom lit les commentaires du fichier en entrée et les écrit dans un fichier de sortie. Pour mettre en forme le contenu dans le fichier de sortie, des balises sont disponibles et doivent être mise dans les commentaires multilignes du fichier de code.

En-tête du fichier

Pour générer l'en-tête du document, il faut faire un commentaire multiligne comme celui suivant :

```
/**
 * @title Nom du fichier
 * @docauthor Votre nom
 * @date la date de création du fichier
 * @make
 */
```

Les balises `@title`, `@docauthor` et `@date` peuvent être mise dans n'importe quel ordre. Il n'y a pas de format particulier pour la date. Si la balise `@date` est manquante, la date du jour sera ajouté à l'en-tête du document.

Structures de données

Pour faire la documentation d'une structure de données, plusieurs balises sont disponibles :

- `@struct` : Nom de la structure
- `@desc` : Description de la structure
- `@author` : Nom de l'auteur de la structure (inutile si le fichier a été rédigé par une seule personne)
- `@field` : Description d'un champ de la structure

L'utilisation de ces balises se fait comme dans l'exemple suivant (en langage c) :

```
/**
 * @struct MaStruct
 * @author Moi (Facultatif)
 * @desc Description de la structure
 * @field premierChamp (int) : description du champ
 * @field secondChamp (float) : description du champ
 */
typedef struct MaStruct
{
    int premierChamp;
    float secondChamp;
}MaStruct;
```

Fonctions

Pour la documentation des fonctions, en plus des balises `@desc` et `@author` vues précédement, les balises suivantes sont disponibles :

- `@fn` : Nom de la fonction
- `@param` : Description d'un paramètre de la fonction
- `@return` : Description de la valeur de retour de la fonction

Les balises `@bcode` (balise ouvrante) et `@ecode` (balise fermante) permettent d'ajouter le prototype de la fonction avec la coloration syntaxique à la documentation (c'est facultatif mais plus joli). Toute les balises sont à utiliser comme dans l'exemple qui suit (en PHP) :

```php
/**
 * @fn maFonction
 * @desc Description de la fonction
 * @author Nom de l'auteur (facultatif)
 * @param $premierParametre (int) : description
 * @param $secondParametre (int) : description
 * @return (float) : description du retour de la fonction
 * @bcode
 */
function maFonction(int $premierParametre, int $secondParametre):float;
/**
 * @ecode
 */
```

Spécificitées liées à l'orienté objet (classes, méthodes, attributs)

Pour la programmation orientée objet, trois balises viennent s'ajouter à toutes celles que nous avons vu :

- `@class` : Le nom de la classe
- `@attribute` : Le nom de l'attribut
- `@method` : Le nom de la méthode

Ces balises s'utilisent avec les autres comme dans l'exemple suivant (en Java) :

```java
/**
 * @class MaClasse
 * @author Moi
 * @desc Description de la classe
 */
public class MaClasse{

    /**
     * @attribute
     * @desc Rôle de l'attribut
     */
    private int attribut;
}
```

Pour les commentaires avant une méthode, c'est exactement la même suite de balises que pour les fonctions, il suffit juste de remplacer la balise `@fn` par la balise `@method`.