Générateur de documentation

Nicolas LE GUERROUÉ

juin 2020

Table des matières

Contact pour l'information

Document réalisé en Latex par Nicolas Le Guerroué
pour un usage du générateur de documentation sur le site electron-56
Telephone : 06.20.88.75.12
E-mail : nicolasleguerroue@gmail.com
Version du 13 septembre 2020

Section 1

Introduction

Cet article a pour but de détailler la génération d'une documentation sur le site http://electron-56.ddns.net

1.1 Principe

La documentation repose sur des mot-clés précis qui permettent de définir les propriétés de la documentation. Ces mot-clés seront appelés **balises**.
Certaines balises permettront de délimiter du contenu de code tandis que d'autres permettrons de stocker des informations de type, de description, etc...

Chaque balise commence obligatoirement avec le caractère @ (arobase) et doit être considérée comme un commentaire dans le langage à documenter.

Voici un exemple de syntaxe :

```
@author Auteur
@language Langage
@lv version_du_langage
@version version_du_projet
@date date_du_projet
```

1.2 Remarques importantes

Ce modèle de documentation présenté est valable uniquement pour les langages proposant des fonctions et procédures ayant un nom, le tout encapsulé dans une classe. Les balises de classes et de méthode ne seront pas valide pour documenter un fichier

CSS.[1] Il est obligatoire de limiter à une classe chaque fichier analysé. Un fichier qui contiendrait deux classes soulèverait une erreur de la part du site.

1. Pour une documentation CSS, se reporter ici

Section 2

Les types de balise

Il existe deux types de balises :

1. les balises **uniques**
2. les balises **de classe et méthode (ou fonctions)**

Section 3

Les balises uniques

Dans le cas des balises uniques, ces balises peuvent être situées **n'importe où dans le fichier source**, à condition que les balises ne soit pas exécutées (ou interprétées) comme du code.

3.1 Présentation des balises

1.
```
@project Nom du projet
```

Le nom du projet

2.
```
@author Auteur
```

Le ou les auteurs

3.
```
@language Langage
```

Le langage

4.
```
@lv Version du langage
```

La version du langage

5.
```
@date XX/XX/XXXX
```

La date du projet

6.
```
@class Nom de la classe
```

Le nom de la classe

7. La description du projet

   Il est possible de faire une description sur plusieurs ligne en ajoutant une balise @presentation à chaque nouvelle ligne.

```
@presentation Suite de la description du projet
@presentation Suite de la description du projet
```

3.2 Gestion de l'héritage

Il est possible d'afficher les méthodes de la classe parent dans la documentation. Pour cela, dans la classe Fille, il faut rajouter la balise **@parent** avec comme argument le nom du fichier comportant la classe Mère.

Cette déclaration se fait n'importe où dans le fichier.

3.3 Exemple

Voici un exemple complet de l'utilisation des balises uniques (Exemple avec le langage Python)

```
@project générateur de fichier LTSpice
@author Nicolas LE GUERROUE
@language PYTHON
@lv 3.7
@version 1.0
@date 05 juin 2020
@presentation La bibliothèque LTSpice permet de générer des fichiers
   LTSpice en commandes Python.
@presentation L'unité de placement des composants est une valeur entiè
   re qui, unitairement, représente le coefficient commun de taille
   des composants LTSpice
@presentation Le curseur se déplacant de 16 en 16, cette unité de
   mesure introduite par la bibliothèque permet de toujours bien
   placer les composants pour qu'ils soient
```

```
@presentation accessibles par le curseur
@class LTSpice
"""
```

3.4 Levée d'avertissement

Une absence constatée d'une des balises soulèvera un avertissement de la part du site. La documentation va néanmoins s'afficher sur le site, sans les informations absentes.

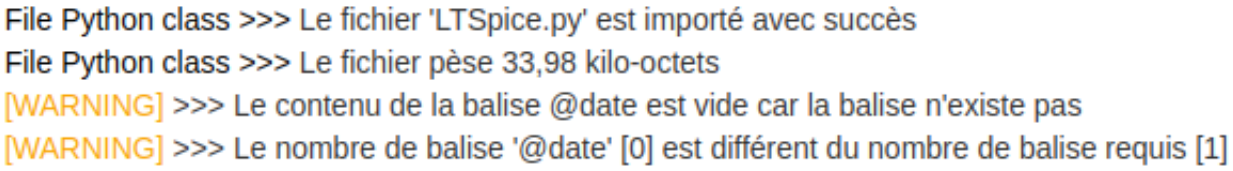

FIGURE 3.1 – Levée d'erreur avec une balise @date absente

Section 4

Les balises de classe et de méthode

Pour les balises de classe et de méthode, ces balises doivent être situées **entre une balise @begin et une balise @end.**

4.1 Présentation des balises

1. Le **nom de la méthode ou de la fonction**

```
@method Nom_de_la_methode
```

nom de la méthode ou de la fonction

Pour documenter un constructeur, il faut remplacer **@method** par **@constructor**

2. La **portée de la méthode ou fonction**

A ce propos, 3 portées sont disponibles :

(a) portée **publique**
La méthode est accessible dès lors de l'instanciation de la classe.

```
@range public
#ou bien
@range publique
```

Portée publique

Il est possible de mettre d'autres variantes du moment que le contenu de la balise contienne au moins le mot 'publi'

(b) portée **protégée**
La méthode est accessible en interne de la classe et par héritage.

```
@range protected
#ou bien
@range protégé(e)
```

Portée protégée

Il est possible de mettre d'autre variante du moment que le contenu de la balise contienne au moins le mot 'prote'

(c) portée **privée**
La méthode est accessible uniquement au sein de la classe.

```
@range private
#ou bien
@range privé(e)
```

Portée privée

Il est possible de mettre d'autre variante du moment que le contenu de la balise contienne au moins le mot 'priv'

3. La **description de la méthode**
Le contenu de la description de la méthode doit être contenu entre les balises @des et @sed

```
@des
Description de la méthode
Suite de la description de la méthode
Suite de la description de la méthode
@sed
```

Description

4. Les **arguments de la méthode** Pour déclarer un paramètre attendu, il convient de mettre une balise @input pour chaque argument. La ligne de balise @input doit contenir une balise de type.
Quelques types d'arguments [1] :

   (a) **@integer** - Entier
   (b) **@string** - Chaine de caractère
   (c) **@float** - Nombre décimal (ou @double)
   (d) **@list** - Liste

1. Tous les types pris en charge sont situés en annexes (Types supportés)

Enfin, on ajoute la description de l'argument

```
@input @integer Nombre pris en argument
@input @string Chaine de caractère prise en argument
```

Arguments

L'ordre des balises est sans importance, cependant, dans un souci de lecture, il convient de mettre la balise @input en première.

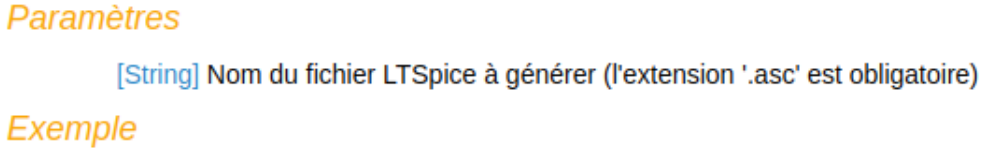

FIGURE 4.1 – Un type @string faisant parti des types autorisés

Lorsque une balise de type n'est pas reconnu, le nom de la balise va s'afficher dans la documentation afin de mettre en valeur le type choisi (type erroné par exemple)

En cas d'erreur, la documentation va afficher ceci :

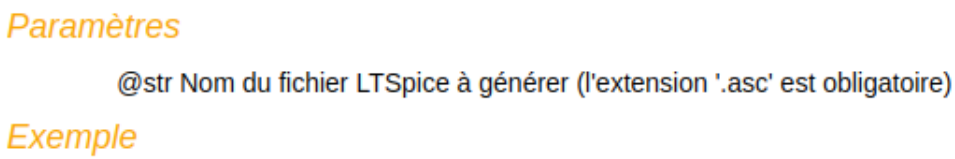

FIGURE 4.2 – Un type @str ne faisant parti des types autorisés

Lorsque il n'y a aucun argument, ne mettez pas de balise @input.

5. Les **types de retour**
   Pour déclarer un type de retour attendu, il convient de mettre une balise **@type_return** puis de mettre une balise de type.
   Lorsqu'il n'y a pas de type de retour, précisez le type comme **@none**

```
@type_return @integer
```

Retour

6. Les **descriptions de retour**
   Pour expliquer un retourde méthode, il convient de mettre une balise **@return** puis de mettre l'explication.
   Lorsqu'il n'y a pas de type de retour, précisez que la fonction ne retourne rien **@none**

```
@return False en cas de succès, True en cas d'erreur
```

Valeur de retour

7. Les **exemples de code**
   Il est possible d'ajouter un éxemple d'utilisation de la méthode. Pour cela, il suffut d'ajouter la balise **@example** puis le code associé (sur la même ligne, pas de retour à la ligne).

```
@example x = ltpice.getX()
```

Exemple

8. Le **code intégral de la méthode**
   Afin de pouvoir récuperer le code de la méthode, il convient de placer le contenu de la méthode entre les balises **@code** et **@end**

4.2 Exemple

```
"""
@begin
@method getX
@range public
@des
Cette méthode permet de retourner la position courante en x du
 curseur dans le fichier.
@sed
@type_return @integer
@return La position du curseur en unité standard (unité=16)
@example position_x = ltspice.x()
@code
"""
def getX(self):
  return int(self.__x/16)
"""
@end
"""
```

4.3 Levée d'avertissement

Une absence constatée d'une des balises soulèvera un avertissement de la part du site. La documentation va néanmoins s'afficher sur le site, sans les informations absentes. **Il faut être conscient qu'une absence de balise va impacter la suite de la documentation, des décalages dans les noms, arguments et autres risquent d'apparaître.**

```
File Python class >>> Le fichier 'LTSpice.py' est importé avec succès
File Python class >>> Le fichier pèse 33,9 kilo-octets
[WARNING] >>> Le nombre de balise '@return' [26] est différent du nombre de balise '@method' [25]
[WARNING] >>> Le nombre de balise '@type_return' [26] est différent du nombre de balise '@method' [25]
```

FIGURE 4.3 – Deux balises manquantes

Section 5

Annexe

5.1 Types supportés

Voici la liste des types pris en charges.

1. caractère - **@char**
2. caractère non signé- **@uchar**
3. Chaîne de caractère - **@string**
4. Entier signé - **@integer**
5. Entier non signé - **@uinteger**
6. Booléen - **@bool**
7. Nombre décimal - **@double** ou **@float**
8. Type automatique (C++)- **@auto**
9. Entier signé sur 8 bits - **@int8_t**
10. Entier non signé sur 8 bits - **@uint8_t**
11. Entier signé sur 16 bits - **@int16_t**
12. Entier non signé sur 16 bits - **@uint16_t**
13. Vecteur (C++) - **@vector**
14. Tableau (C++, PHP) - **@array**
15. Iterateur (C++) - **@iterator**
16. Pointeur sur caractère - **@*char**

17. Pointeur sur caractère non signé- **@*uchar**
18. Pointeur sur chaîne de caractère - **@*string**
19. Pointeur sur entier signé - **@*integer**
20. Pointeur sur entier non signé - **@*uinteger**
21. Pointeur sur booléen - **@*bool**
22. Pointeur sur nombre décimal - **@*double** ou **@*float**
23. Pointeur sur entier signé sur 8 bits - **@*int8_t**
24. Pointeur sur entier non signé sur 8 bits - **@*uint8_t**
25. pointeur sur entier signé sur 16 bits - **@*int16_t**
26. Pointeur sur entier non signé sur 16 bits - **@*uint16_t**
27. Pointeur sur Iterateur (C++) - **@*iterator**
28. Référence sur caractère - **@&char**
29. Référence sur caractère non signé- **@&uchar**
30. Référence sur chaîne de caractère - **@&string**
31. Référence sur entier signé - **@&integer**
32. Référence sur entier non signé - **@&uinteger**
33. Référence sur booléen - **@&bool**
34. Référence sur nombre décimal - **@&double** ou **@&float**
35. Référence sur entier signé sur 8 bits - **@&int8_t**
36. Référence sur entier non signé sur 8 bits - **@&uint8_t**
37. Référence sur entier signé sur 16 bits - **@&int16_t**
38. Référence sur entier non signé sur 16 bits - **@&uint16_t**
39. Référence sur Iterateur (C++) - **@&iterator**