



Manuel du développeur de DocGen

Auteur(s) : Florian Legendre



Légendes et Abréviations utilisées

```
1 Ceci est du code source.  
2 Selon les langages, différents mots seront colorés selon  
3 si ce sont des mots clefs ou non (comme int, char, etc.).
```

Listing 1 – Exemple de code source

```
Ceci est un formatage automatique Latex d'un texte copié-collé  
directement depuis un terminal Bash ayant valeur de capture  
d'écran. La coloration correspond à une coloration quelconque  
d'un terminal Bash (les chemins étant habituellement coloré et  
le nom de l'utilisateur aussi comme crex@crex:~$ ...)
```

Listing 2 – Exemple d'une pseudo capture d'écran Bash

Table des matières

I	Cahier des Charges	3
II	Spécifications Fonctionnelles	4
1	Fonctionnement global de DocGen	5
1.1	Diagramme de classe	5
2	Génération du Header	6
3	Génération des Index	7
3.1	Index hiérarchiques	7
3.1.1	Définition	7
3.1.2	Solution 1 : Traitement d'une liste de chemins absolus	8
	Description de l'algorithme	8
	Zoom sur l'étape 3 (fonction globalIndexArrMake())	12
	Bilan	13
4	Génération des Notices	14
5	Génération des Pages de Manuel	15
III	Bibliographie et Glossaires	16
IV	Annexes	17

Première partie

Cahier des Charges

Deuxième partie

Spécifications Fonctionnelles

Chapitre 1

Fonctionnement global de DocGen

Diagramme de classe

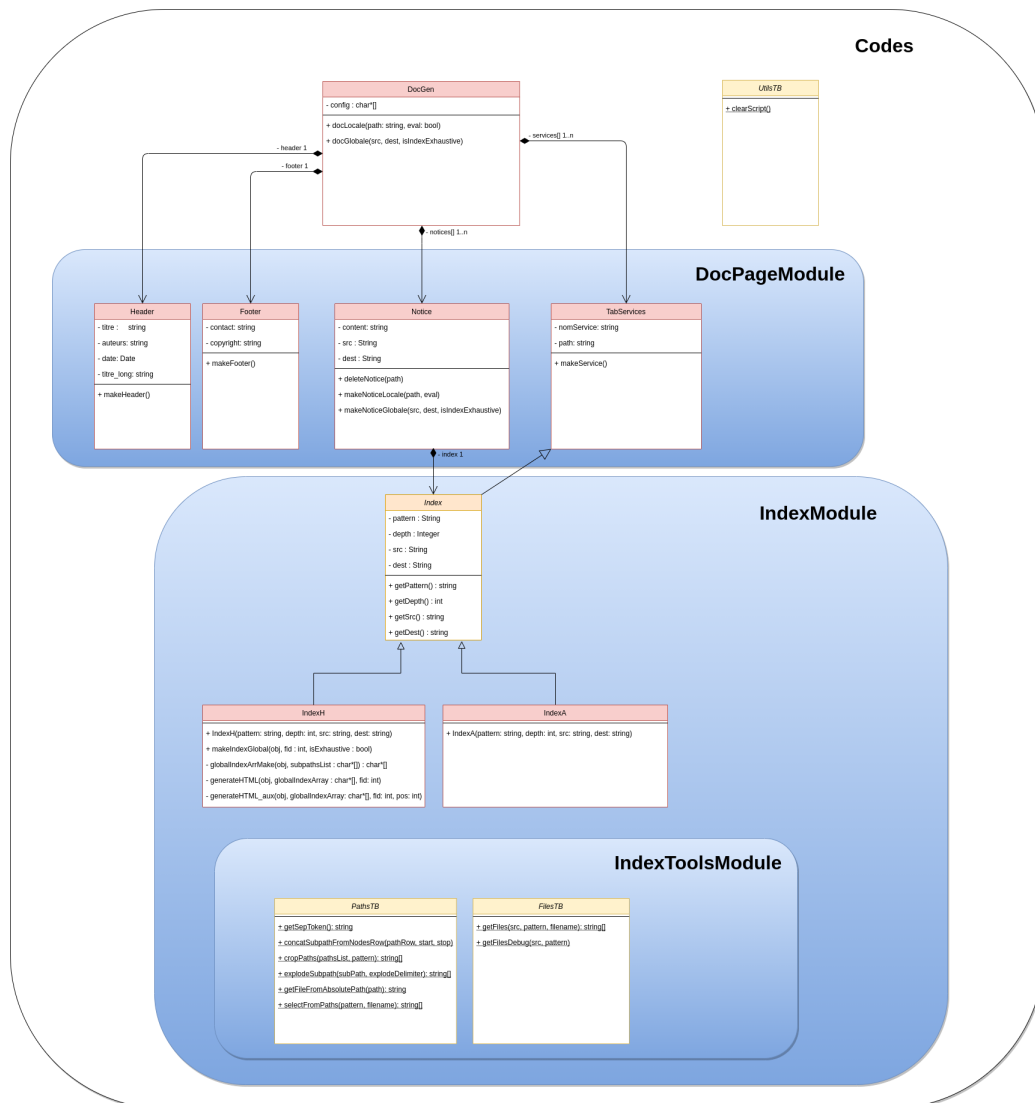


FIGURE 1.1 – Diagramme de classe de DocGen

Chapitre 2

Génération du Header

Chapitre 3

Génération des Index

Index hiérarchiques

3.1.1 Définition

Un index hiérarchique ou arborescent est une forme d'indexation où les entrées sont contenues dans d'autres entrées, etc. reflétant ainsi la généralisation (on remonte dans l'arbre) ou la spécification (on descend dans l'arbre) des entrées les unes par rapport aux autres.

Ce type d'index est appelé *indexT* dans le code en référence à l'anglais *Tree Index*.

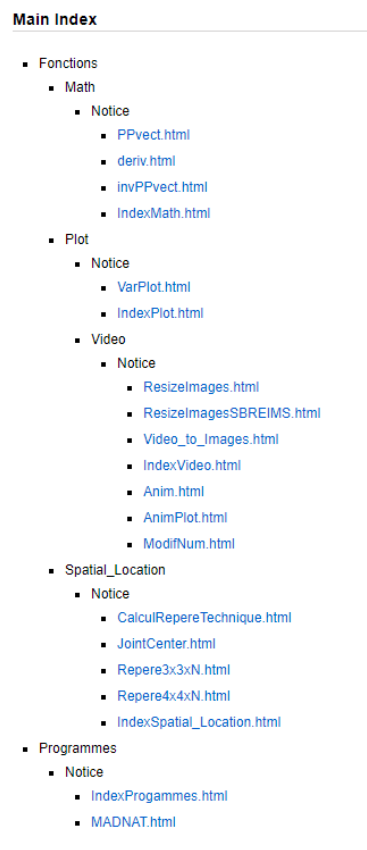


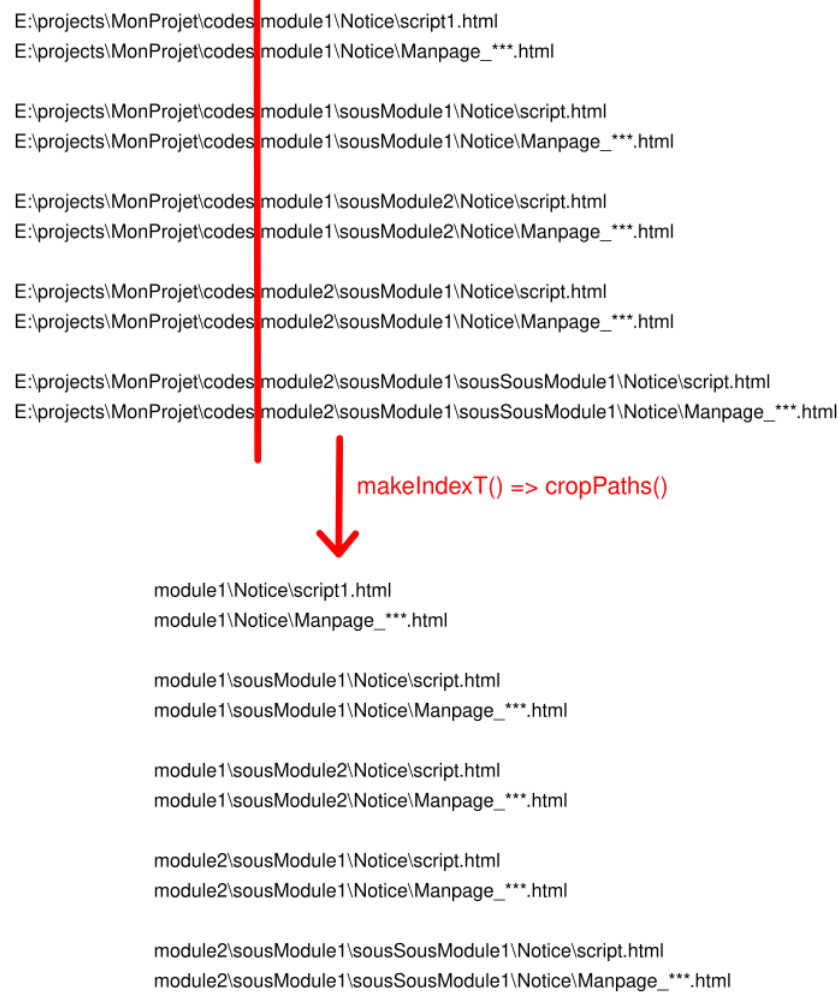
FIGURE 3.1 – Exemple d'indexation hiérarchique

3.1.2 Solution 1 : Traitement d'une liste de chemins absolus

Description de l'algorithme

Étape 1

On tronque de la liste des chemins récoltés la partie correspondant à la racine où est effectué l'index et on en profite pour mettre les chemins tronqués dans un tableau de chaînes de caractères.



```
E:\projects\MonProjet\codes\module1\Notice\script1.html
E:\projects\MonProjet\codes\module1\Notice\Manpage_***.html

E:\projects\MonProjet\codes\module1\sousModule1\Notice\script.html
E:\projects\MonProjet\codes\module1\sousModule1\Notice\Manpage_***.html

E:\projects\MonProjet\codes\module1\sousModule2\Notice\script.html
E:\projects\MonProjet\codes\module1\sousModule2\Notice\Manpage_***.html

E:\projects\MonProjet\codes\module2\sousModule1\Notice\script.html
E:\projects\MonProjet\codes\module2\sousModule1\Notice\Manpage_***.html

E:\projects\MonProjet\codes\module2\sousModule1\sousSousModule1\Notice\script.html
E:\projects\MonProjet\codes\module2\sousModule1\sousSousModule1\Notice\Manpage_***.html
```

makeIndexT() => cropPaths()

```
module1\Notice\script1.html
module1\Notice\Manpage_***.html

module1\sousModule1\Notice\script.html
module1\sousModule1\Notice\Manpage_***.html

module1\sousModule2\Notice\script.html
module1\sousModule2\Notice\Manpage_***.html

module2\sousModule1\Notice\script.html
module2\sousModule1\Notice\Manpage_***.html

module2\sousModule1\sousSousModule1\Notice\script.html
module2\sousModule1\sousSousModule1\Notice\Manpage_***.html
```

FIGURE 3.2 – Suppression de la partie commune inutile

Étape 2

Si le paramètre *isExhaustive* est à *false* on supprime également de l'indexation les entrées qui ne sont pas des pages de manuel comme illustré ici :

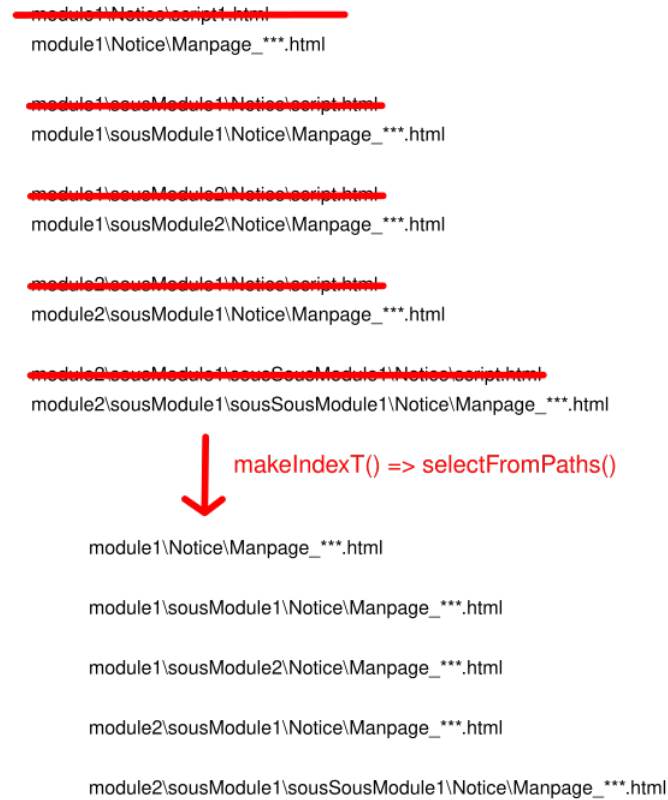


FIGURE 3.3 – Suppression des entrées qui ne sont pas des pages de manuel

Étape 3

Chaque chemin absolu est ensuite traité par la méthode *globalIndexArrMake(obj, subpathsList)* qui se charge d'en faire un tableau de noeuds¹. Les chemins absolus sont reconstitués en reconcaténant la partie tronquée à l'étape 1.

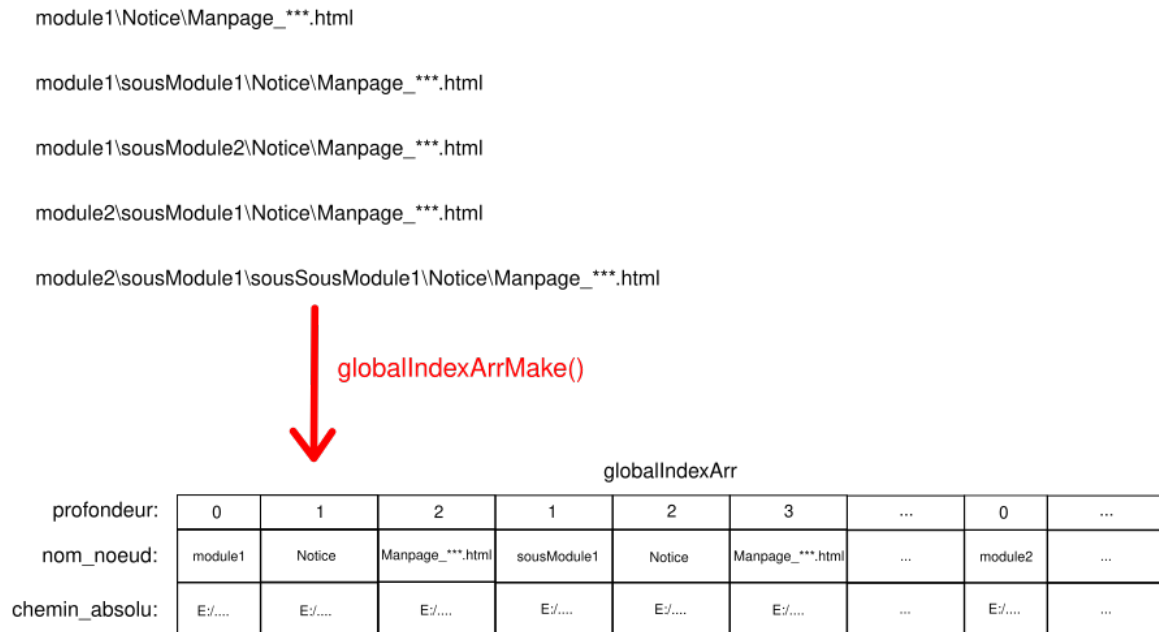


FIGURE 3.4 – Transformation des chaînes de caractères en tableaux de noeuds

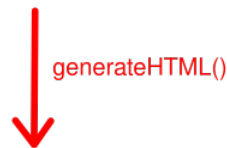
1. Une structure de donnée serait probablement plus appropriée ici...

Étape 4

Une fois le tableau de noeud construit nous avons toutes les informations pour traduire celui-ci sous format HTML en écrivant directement dans le fichier voulu. Pour ce faire on tire profit de la récursivité dès lors qu'on s'enfonce ou qu'on remonte des profondeurs. On utilise en revanche une boucle pour le cas où on reste à la même profondeur.

Le résultat est un ensemble de listes HTML imbriquées comme illustré ici :

	globalIndexArr								
profondeur:	0	1	2	1	2	3	...	0	...
nom_noeud:	module1	Notice	Manpage_***.html	sousModule1	Notice	Manpage_***.html	...	module2	...
chemin_absolu:	E:/...	E:/...	E:/...	E:/...	E:/...	E:/...	...	E:/...	...



generateHTML()

Manpage_Global.html

```
[...]  
  
<ul>  
  <li> module1  
    <ul>  
      <li> Notice  
        <ul>  
          <li> <a href = "file:///E:/..."> Manpage_***.html </a> </li>  
        </ul>  
      </li>  
      <li> sousModule1  
        <ul>  
          <li> Notice  
            <ul>  
              <li> <a href = "file:///E:/..."> Manpage_***.html </a> </li>  
            </ul>  
          </li>  
          [...] </li>  
        </ul>  
      </li>  
      [...] </li>  
    </ul>  
  </li>  
  <li> module 2  
    [...] </li>  
</ul>  
  
[...]
```

FIGURE 3.5 – Écriture du tableau de noeuds dans le fichier désigné sous format HTML

Zoom sur l'étape 3 (fonction globalIndexArrMake())

Cette fonction, avec celle de génération du code HTML, est probablement la fonction la plus difficile à comprendre de DocGen et donc je tiens ici à offrir une description détaillée et imagée de son fonctionnement. Tout d'abord, je rappelle ici le code source :

```
1 function globalIndexArray = globalIndexArrMake(obj, subpathsList)
2 % Algorithm initialization %
3 appendIdx = 1;
4 nbSubpaths = numel(subpathsList);
5 precPathRow = PathsTB.explodeSubpath(subpathsList{1}{1}, PathsTB.
    setgetVar);
6 nbNodes = numel(precPathRow);
7 for i=1:nbNodes
8     nodePath = [obj.getSrc() PathsTB.setgetVar ...
9     PathsTB.concatSubpathFromNodesRow(precPathRow, 1, i)];
10    globalIndexArray{appendIdx} = [i, precPathRow(i), nodePath];
11    appendIdx = appendIdx + 1;
12 end
13
14 % Global index making %
15 for i=2:nbSubpaths
16     currentSubpathRow = PathsTB.explodeSubpath(subpathsList{i}{1},
17     PathsTB.setgetVar);
18     nbNodes = numel(currentSubpathRow);
19
20     updatePrecPath = false;
21     for j=1:nbNodes
22         if j > numel(precPathRow) || ~strcmp(precPathRow(j),
23         currentSubpathRow(j))
24             updatePrecPath = true;
25         end
26
27         if updatePrecPath
28             nodePath = [obj.getSrc() PathsTB.setgetVar ...
29             PathsTB.concatSubpathFromNodesRow(
30                 currentSubpathRow, 1, j)];
31             globalIndexArray{appendIdx} = [j,
32             currentSubpathRow(j), nodePath];
33             precPathRow(j) = currentSubpathRow(j);
34             appendIdx = appendIdx + 1;
35         end
36     end
37 end
38 end
```

Listing 3.1 – Code source de la fonction de transformation de liste de chemins absolus en tableaux de noeuds

Bilan

Chapitre 4

Génération des Notices

Chapitre 5

Génération des Pages de Manuel

Troisième partie

Bibliographie et Glossaires

Quatrième partie

Annexes