Les générateurs de documentation

Doxygen, Ccdoc, Javadoc, Phpdoc

T.Odorico A.Gouyon

Université des sciences et des lettres

18 avril 2017

Générateur de documentation

Générateurs de Documentation

Qu'est-ce donc ?

- Un générateur de documentation est un outil de programmation qui crée de la documentation destinée aux programmeurs.
- Pour gérer ces documentations, le générateur se base généralement sur des codes sources commentés d'une certaine façon et dans certains cas également sur des fichiers binaires.

Qu'est-ce donc ?

- La documentation générée peut être hautement technique, et est principalement utilisée pour définir et expliquer les structures de données et les algorithmes.
- Par exemple, on peut utiliser cette documentation pour expliquer que la variable name se réfère au premier et au dernier nom d'une personne.
- Il est important pour les documents sur le code d'être précis, mais pas non plus verbeux à un point tel qu'il serait difficile de les maintenir.

Qu'est-ce donc ?

- Les générateurs de documentation tels doxygen ou javadoc génèrent automatiquement la documentation à partir du code source.
- Les documents sur le code sont souvent organisés dans le style d'un guide de référence, ce qui permet à un programmeur de localiser rapidement une fonction ou une classe quelconque.

Qu'est-ce donc ?

- L'avantage d'un générateur de documentation (à partir du code source) est la proximité du code source avec sa documentation codée sous forme de commentaires.
- Le programmeur peut alors l'écrire en se référant à son code, et peut utiliser les mêmes outils que ceux qu'il a utilisés pour développer le code source, pour faire la documentation.

Qu'est-ce donc ?

- Cela rend beaucoup plus facile la mise à jour de la documentation.
- Bien sûr, l'inconvénient est que seuls les programmeurs peuvent éditer cette sorte de documentation.

Exemple avec Javadoc

```
/**
 * Valider un mouvement de jeu d'échecs.
 *
 * @param colonneDepart  Colonne de la case de départ de la pièce à
déplacer.
 * @param ligneDepart    Ligne   de la case de départ de la pièce à
déplacer.
 * @param colonneArrivee Colonne de la case de destination.
 * @param ligneArrivee   Ligne   de la case de destination.
 * @return               vrai(true) si le mouvement d'échec est valide
ou faux(false) si invalide.
 */
 boolean estUnDéplacementValide(int colonneDepart,  int ligneDepart,
                                int colonneArrivee, int ligneArrivee)
 {
     ...
 }
```

Exemple avec Doxygen

```
/**
 * Valider un mouvement de jeu d'échecs.
 * @return               Validité du mouvement d'échec.
 * @retval vrai(true)    Mouvement d'échec valide.
 * @retval faux(false)   Mouvement d'échec invalide.
 */
int estUnDéplacementValide
(
    //!  Colonne de la case de départ de la pièce à déplacer.
    int colonneDepart,
    //!  Ligne de la case de départ de la pièce à déplacer.
    int ligneDepart,
    //!  Colonne de la case de destination.
    int colonneArrivee,
    //!  Ligne   de la case de destination.
    int ligneArrivee)
{
    //   ...
}
```

Générateurs de Documentation

Caractéristiques

- Javadoc permet de creer une documentation d'api à partir des commentaires contenu dans son code java
- Pour que les blocs de commentaires soit reconnu par javadoc il doivent commencer par un "/**" et finir par un "*/", de plus chaque ligne commence par un "*".
- Il est possible de classifier les informations inclues dans les commentaires grâce a des tags specials

Caractéristiques

- "@author" suivi de "bob" indiquera à javadoc que bob est l'auteur de cette partie du code et @version indique la version du code.

Example

```
/**
* @autor bob
* @version 1.42
*/
public class Kebab
...
```

Générateurs de Documentation

Listes des tags et leurs fonctions

- @author Nom du développeur
- @deprecated Marque la méthode comme dépréciée. Certains IDEs créent un avertissement à la compilation si la méthode est appelée.
- @exception Documente une exception lancée par une méthode — voir aussi @throws.
- @param Définit un paramètre de méthode. Requis pour chaque paramètre.
- @return Documente la valeur de retour. Ce tag ne devrait pas être employé pour des constructeurs ou des méthodes définies avec un type de retour void.

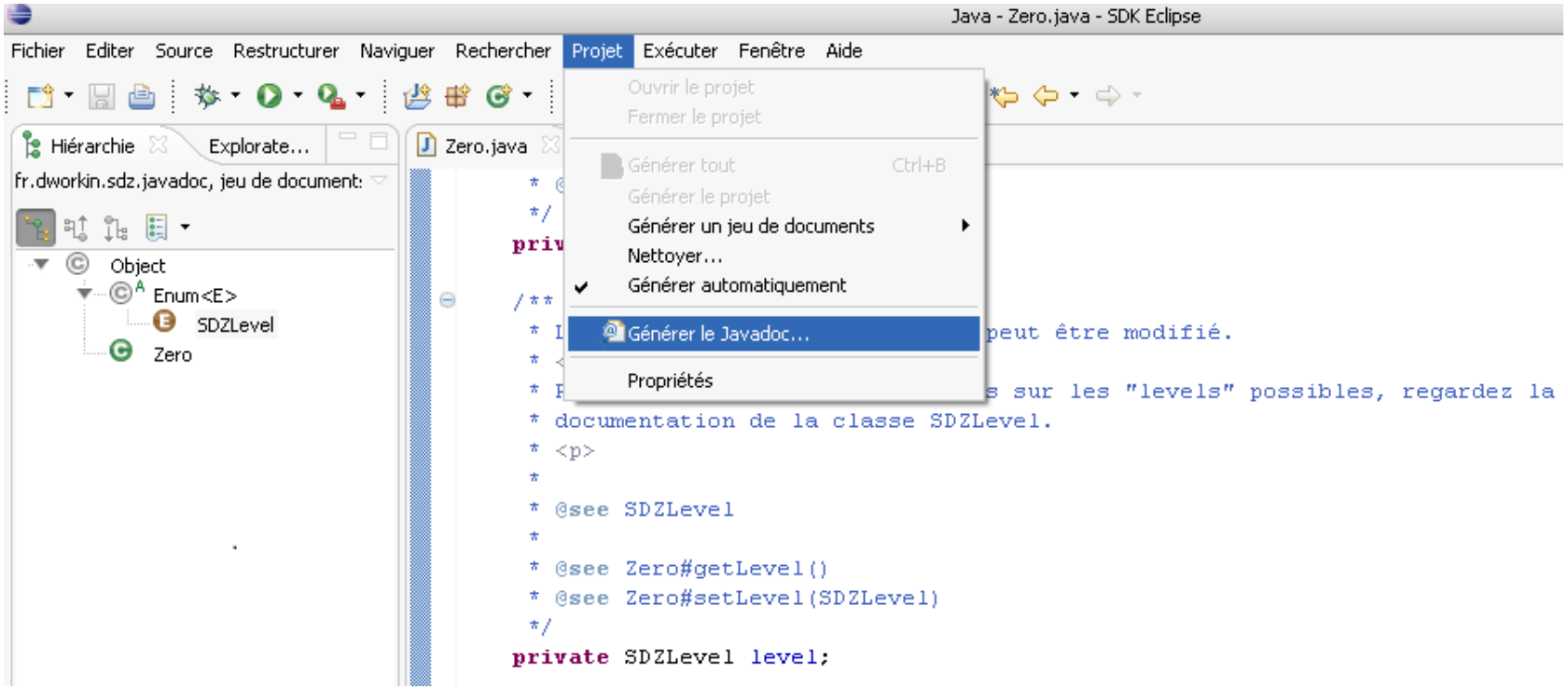
Listes des tags et leurs fonctions

- @see Documente une association à une autre méthode ou classe.
- @since Précise à quelle version de la SDK/JDK une méthode a été ajoutée à la classe.
- @throws Documente une exception lancée par une méthode. Un synonyme pour @exception disponible depuis Javadoc 1.2.
- @version Precise la version du code

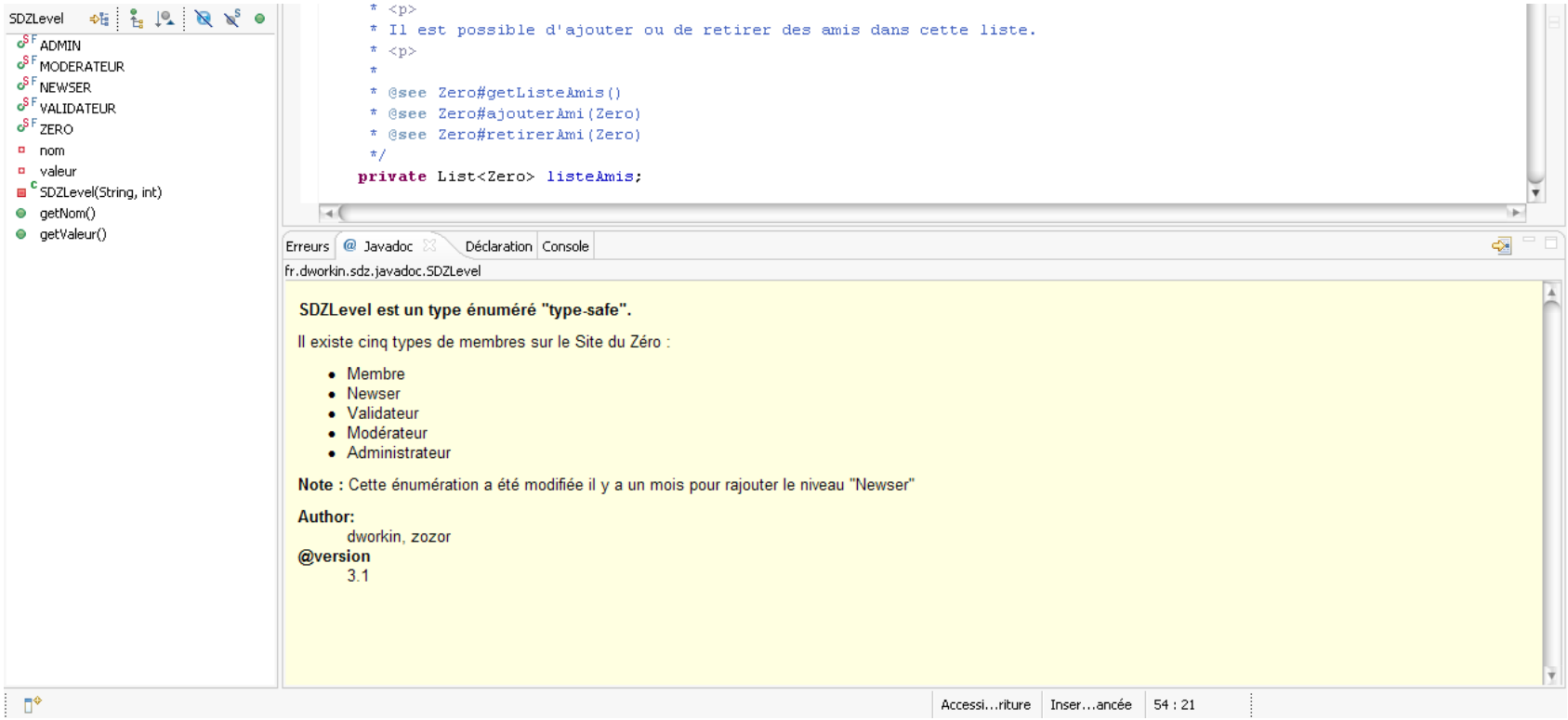
Generer la documentation

- En general les EDI integrent une fonction qui permet à l'uttilisateur de generer la documentation facilement.
- Pour de la programmation en java on pourrai uttiliser l'EDI Eclipse aller dans "projet" puis generer la doc ;

Generer la documentation



Generer la documentation



Generer la documentation

- On peut aussi uttiliser une ligne de commande avec javaDoc

Example

```
javadoc fichierSource.java -d doc
```

- Cette ligne va generer la documentation de fichierSource dans le dossier doc

Générateurs de Documentation

Generer la documentation

- Ccdoc et PHPdoc sont utillisées pour generer la documentation respectivement pour C/C++ et pour le PHP, leurs principe est exactemement le même que pour javadoc et reperent les commentaires de la meme maniere, (/**, *, @tag, */) ils sont d'ailleurs fortement inspirer de celui-ci.

Ccdoc

- On peut generer la documentation avec Ccdoc sur linux avec une ligne de commande :

Example

```
mkdir webdocs
ccdoc -db documentation.db fichierSource.h -html webdocs/
```

PHPdoc

- Puis sur PHPdoc on peut faire de même :

Example

```
phpdoc -d dossierDoc -f fichierSource.php
```

Générateurs de Documentation

Mais c'est quoi ?

- A l'instar de Javadoc, Doxygen est un logiciel libre de documentation de code possédant des capacités de génération de documentation à partir du code source d'un programme.
- Pour cela il tient compte de la syntaxe et de la structure du langage du programme ainsi que des commentaires associés à condition que ceux-ci soient écrit dans un format adapté.

Mais c'est quoi ?

- La génération de documentation peut être faite à partir de code dans les langages suivants : C, C++, Java, Objective C, Python, IDL, VHDL et dans une certaine mesure PHP et D.
- Le résultat final est une documentation complète générée en HTML (compressé ou non), LaTeX, RTF, PostScript, PDF avec hyperliens, et XML.

Mais c'est quoi ?

- En permettant l'intégration de la documentation directement dans le code source, Doxygen permet de favoriser la cohérence entre la documentation et le code et de systématiser le comportement des développeurs afin qu'ils documentent le code qu'ils produisent.
- Il est également possible d'extraire de la documentation à partir d'un code source non documenté au préalable, ce qui peut faciliter la compréhension d'un programme dont le code est compliqué.

Générateurs de Documentation

- On va pouvoir voir différentes commandes et leur utilisations pour illustrer le fonctionnement de Doxygen. (Toutes les commandes doivent être précédées d'un \)

a

- Utiliser pour faire ressortir un paramètre dans sa description.

mages/m1.png

bug

- Utiliser pour indiquer un bug dans une fonction ou un morceau de programme.

```
/** \bug La probabilité calculée dépasse parfois 1. */
```

class [] []

- Utiliser pour décrire une classe. Le premier paramètre est obligatoire, les suivants optionnels. Attention toutefois à ce que les paramètres soit corrects, la casse est prise en compte.

```
/** \class Voiture voiture.h "inc/voiture.h" */
class Voiture { };
```

file

- Utiliser pour décrire un fichier. L'attribut doit être exact et fourni avec l'extension.

```
/** \file mon_fichier.c */
```

todo description

- Utiliser pour indiquer ce qu'il reste à faire. La description peut être multiligne.

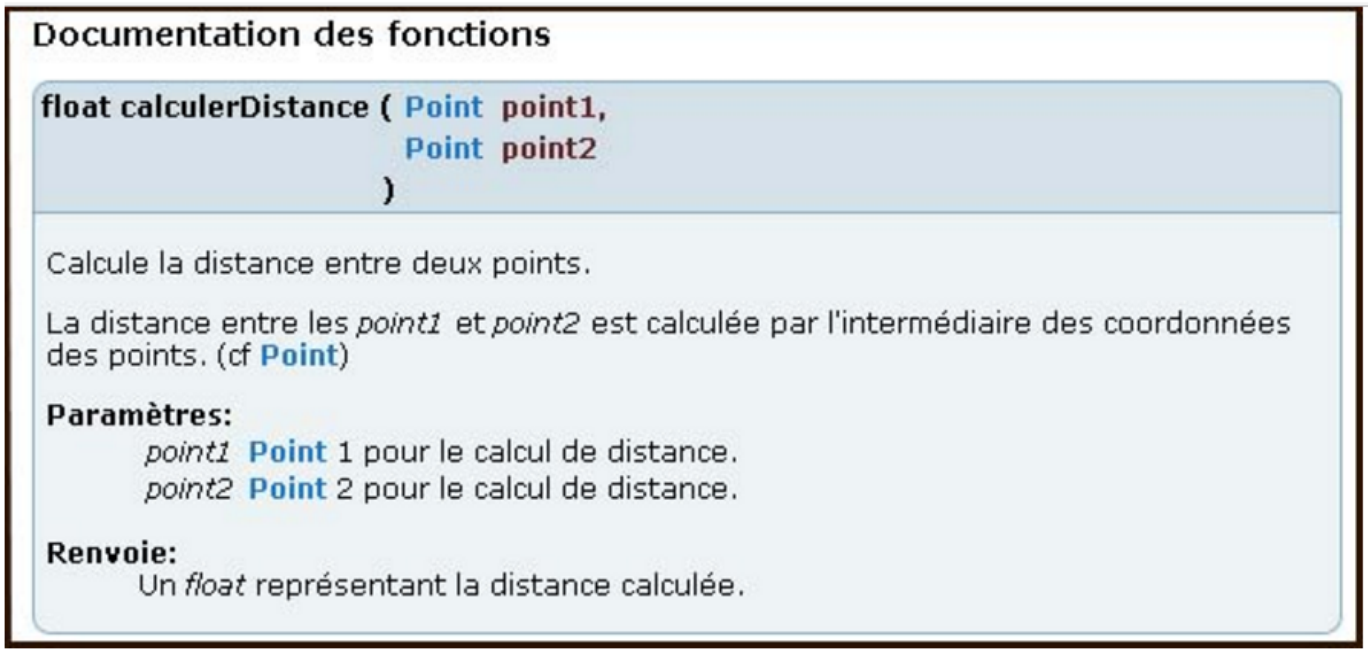
```
/**
 * \todo  Finir l'implémentation des accesseurs de la classe...
 *            ...et encore détails nécessitant plusieurs lignes
 */
```

typedef

- Utiliser pour introduire la description d'un typedef.

```
/**  \typedef  Voiture: la classe voiture  */
```

Exemple de résultat



Documentation des fonctions

float calculerDistance (Point point1,
Point point2
)

Calcule la distance entre deux points.

La distance entre les *point1* et *point2* est calculée par l'intermédiaire des coordonnées des points. (cf **Point**)

Paramètres:

point1 **Point** 1 pour le calcul de distance.
point2 **Point** 2 pour le calcul de distance.

Renvoie:

Un *float* représentant la distance calculée.

Fin

- Merci d'avoir écouté !