Génie Logiciel

Dossier Doxygen

# Introduction

La documentation d’un projet en informatique est primordiale. Une bonne documentation permet d’avoir un code compréhensible et maintenable, celle-ci peut passer uniquement par des commentaires mais ce système ne permet pas d’avoir une vision globale du projet. De plus, les développeurs utilisant une API ou une bibliothèque ont besoin d’une documentation des fonctionnalités sans avoir à consulter les sources de celles-ci. Ces besoins ont menés à la conception d’un outil permettant une meilleure lisibilité des documentations et une normalisation des commentaires de documentation.

Doxygen est un outil de génération de documentation pour du code source annoté. Doxygen est initialement prévu pour la documentation de code source C++ mais il supporte également les sources C, Objective-C, C#, PHP, Java, Python, IDL, Fortran, VHDL, Tcl et D. Le logiciel analyse les commentaires et le code écrit et en extrait la documentation à partir des balises placées dans les commentaires. La documentation peut ensuite être mise en forme selon différents formats :

* HTML pour héberger la documentation en ligne ;
* PDF, LATEX et autres pour une documentation hors-ligne ;
* Pages man pour UNIX.

Doxygen peut également être utilisé pour analyser la structure du code à partir de sources non documentées. Cette fonctionnalité trouve son utilité pour la reprise de projets de taille conséquente puisque Doxygen peut générer automatiquement les diagrammes d’héritage, de collaboration et de dépendance.

Doxygen est sous licence GPLv2, son code source peut donc être utilisé, copié, modifié et redistribué. De ce fait, le logiciel est portable sur la majorité des plateformes actuelles.

Ici, nous verrons la mise en place de Doxygen et son utilisation. Nous aborderons également son fonctionnement et les extensions possibles.

# Installation

Le dépôt des sources de Doxygen est disponible sur <https://github.com/doxygen/doxygen> et les installeurs sur la page <http://www.stack.nl/~dimitri/doxygen/download.html>. Le tutoriel permet une installation simple de Doxygen. Bien que le logiciel puisse être utilisé uniquement en ligne de commande, l’outil Doxywizard permet d’ajouter une interface pour l’édition du fichier de configuration de Doxygen ou plus communément appelé Doxyfile.

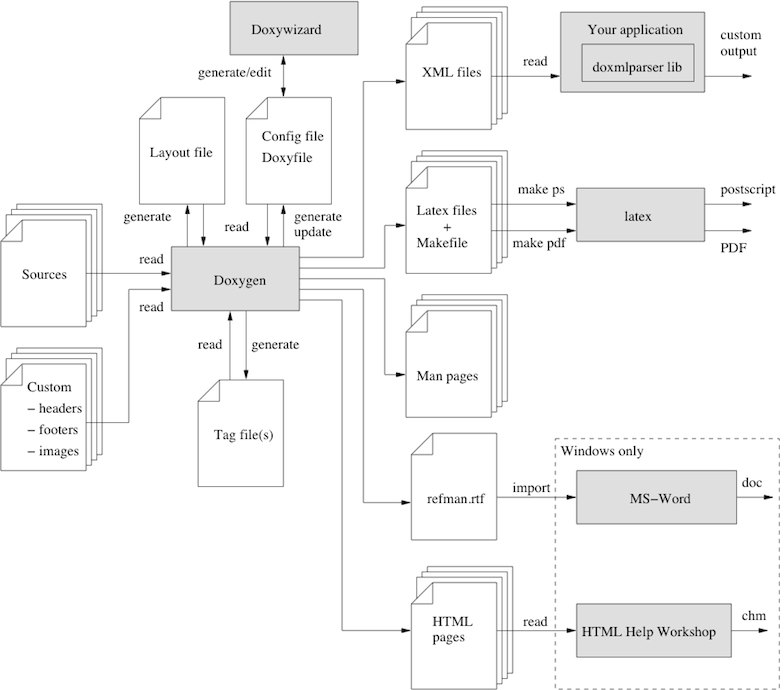


Figure 1 - Flux d'information de Doxygen

# Fonctionnement

L’exécutable doxygen est le cœur de l’outil, il permet de transformer les annotations des sources en différents formats de documentation selon le fichier de configuration ou les arguments passés. Dans cette partie, nous verrons les différents modules et leur rôle dans ces opérations.

## Parsers de configuration

Le rôle des parsers de configuration est de récupérer les paramètres de la ligne de commande ou du fichier de configuration afin de rendre cette configuration exploitable par les autres modules de Doxygen. Les arguments non renseignés sont fixés à une valeur par défaut. Le fichier de configuration permet d’obtenir des informations telles que les sources à analyser, les formats de sorties à utiliser, si les sources non documentées sont à analyser ou encore un fichier de tags à parser. Un tel fichier sera pris en charge par un autre module de chargement de configuration.

Doxygen permettant une configuration avec de très nombreuses options, celles-ci ne seront pas toutes détaillées ici, cependant, une liste exhaustive des arguments possibles est disponible sur <http://www.stack.nl/~dimitri/doxygen/manual/config.html>.

## Parsers de sources

Une fois la configuration chargée, un préprocesseur C va analyser les sources spécifiées dans la configuration, celui-ci s’exécutera sur les fichiers spécifiés dans les #include des sources, permettant ainsi d’avoir une connaissance de l’intégralité de la carte de code à analyser.

Les sources modifiées par le préprocesseur sont ensuite envoyées à un module unique permettant de parser tous les langages compatibles avec Doxygen via l’utilisation d’une machine à états. Le découpage de ce module en des modules spécifiques à des langages est envisagé dans l’avenir de Doxygen.

## Organiseur de données

En sortie de ces modules, Doxygen possède un arbre d’entrées correspondant aux classes, fichiers, namespaces, variables, fonctions, packages, pages et groupes extraits. Le rôle de l’organiseur de données et de construire des dictionnaires catégorisés mais également de créer les relations et les héritages entre les entités de l’arbre.

## Générateurs de sortie

Après la récupération et l’organisation des données, Doxygen génère des sorties dans différents formats. Les générateurs de sorties implémentent tous la classe OutputGenerator. Pour l’instant, un fichier XML peut être généré directement de la sortie de l’organiseur de données. Dans le futur de Doxygen, ce fichier XML pourra être utilisé comme langage intermédiaire pour les générateurs de sorties afin de les détacher encore plus du cœur du logiciel.

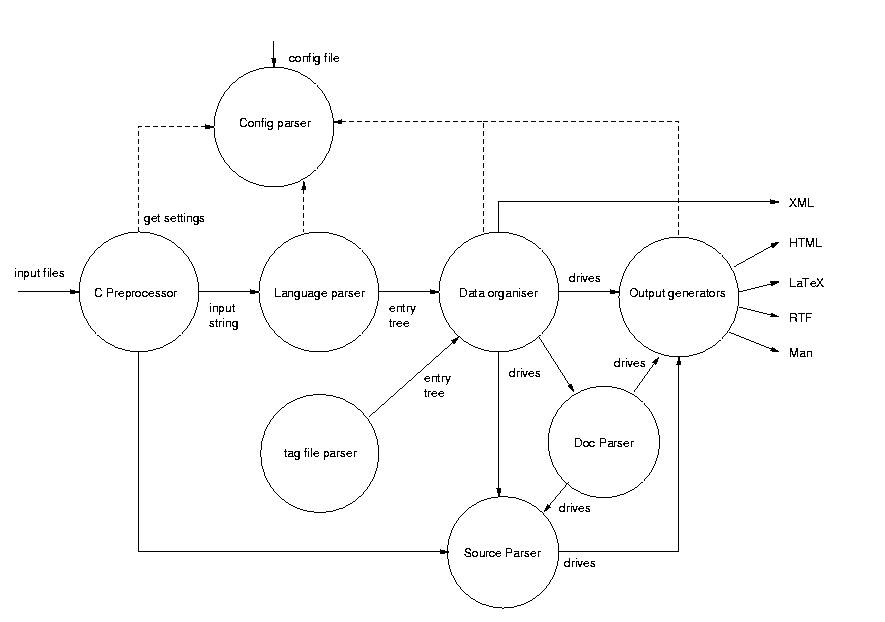


Figure 2 - Structure du noyau de Doxygen

# Création de documentation

## Syntaxe

Doxygen dispose de différentes syntaxes permettant la documentation des sources. Entre autres, l’une d’entre elle supporte la syntaxe de la Javadoc. Doxygen récupère les informations de l’utilisateur dans des blocs de commentaire spécifiques (Voir Figure 3). Ces commentaires documentent soit l’entité se trouvant à leur suite, soit une entité donnée si elle est spécifiée.

/\*\*

\* Bloc de commentaire Doxygen

\*/

/\*!

\* Autre bloc de commentaire Doxygen

\*/

/\*

\* Bloc de commentaire non traité par Doxygen

\*/

/// Ligne de commentaire Doxygen

//! Autre ligne de commentaire Doxygen

// Ligne de commentaire non traitée par Doxygen

Figure 3 - Commentaires Doxygen

## Balises

Doxygen récupère les textes associés aux balises dans les commentaires pour créer la documentation. Ces balises commencent soit par un backslash (\) ou par une arobase (@).

Doxygen possède de nombreuses balises, d’autres peuvent être spécifiées par l’utilisateur via des fichiers de tags. Ici, nous verrons les balises les plus communément utilisées via des exemples d’utilisation.

### Balisage général

Certaines balises peuvent être utilisées sur toutes les entités, celles-ci donnent des descriptions d’ordre général sur les entités. La Figure 4 donne un exemple d’utilisation de ces balises.

/\*!

\* **\brief** Description de la classe

\* **\details** Cette classe est utilisée pour détailler des balises générales

\* **\author** John Doe (Auteur de la classe)

\* **\author** John Smith (Autre auteur de la classe)

\* **\version** 0.5.6b (Version de la classe/du programme)

\* **\date** 1992-2014 (Date de développement du programme)

\* **\pre** Initialiser le système (prérequis d’utilisation)

\* **\bug** Toute la mémoire n’est pas libérée (bug lié à l’utilisation)

\* **\warning** Peut causer des crashs (avertissement lié à l’utilisation)

\* **\copyright** GNU Public Licence **(**copyright d’utilisation de la classe)

\*/

class ExampleClass **{};**

Figure 4 - Exemple d'utilisation des balises générales

### Balisage de fonction et méthodes

Des balises plus spécifiques permettent de documenter des fonctions. Celles-ci donnent des informations sur les paramètres de la fonction et son retour. La Figure 5 donne un exemple de documentation de fonction.

/\*!

\* **\brief** foo Ce texte sera dans la description rapide de la fonction

\* \details Ce texte sera consultable dans la description détaillée

\* **\param** bar Ce texte documente le rôle du paramètre

\* **\param** src[in] Ce paramètre est un paramètre d'entrée de la fonction

\* **\param** dest[out] Ce paramètre est un paramètre de sortie de la fonction

\* **\param** mod[in,out] Ce paramètre est en entrée et en sortie de la fonction

\* **\return** Ce texte documente la sortie de la fonction

\*/

double foo **(**int bar**,** const void **\***src**,** void **\***dest**,** char **\***mod**);**

Figure 5 - Exemple de documentation de fonction

### Balisage de classe et énumération par notation post fixée

Le balisage d’une classe et d’une énumération nécessite parfois l’utilisation d’un balisage post-fixe pour alléger la notation et pouvoir faire la documentation sur la même ligne que la déclaration. La Figure 6 donne un exemple de cette utilisation.

/\*!

\* **\brief** Classe documentée par Doxygen

\*/

class Test

**{**

public**:**

/\*!

\* **\brief** Une énumération documentée

\*/

enum TEnum **{**

TVal1**,** //!< Valeur d'énumération TVal1 en notation post fixée

TVal2**,** //!< Valeur d'énumération TVal2 en notation post fixée

TVal3 //!< Valeur d'énumération TVal3 en notation post fixée

**};**

/\*!

\* **\brief** Constructeur de Test

\* ...

\*/

Test**();**

/\*!

\* **\brief** Destructeur de Test

\* ...

\*/

**~**Test**();**

/\*!

\* **\brief** Méthode publique

\* ...

\* **\param** a Paramètre de méthode

\* **\return** Retour de méthode

\*/

int method**(**int a**);**

int m\_Attribut**;** //!< **\brief** Attribut public en notation post fixée

**};**

Figure 6 - Exemple de documentation de classe et énumération

## Génération de documentation

La configuration du fichier de configuration de Doxygen peut être faite via l’utilitaire DoxyWizard. Celui-ci peut également lancer la génération. Il faut lui spécifier l’emplacement des sources, des répertoires de destinations, choisir les formats de destinations et d’autres options plus spécifiques.

Une de ces options est la génération de diagramme avec l’utilisation du plugin Graphviz/dot. Celui-ci permet une visualisation des héritages entre les différentes classes et d’obtenir un diagramme UML du projet (Voir exemple en Figure 7)

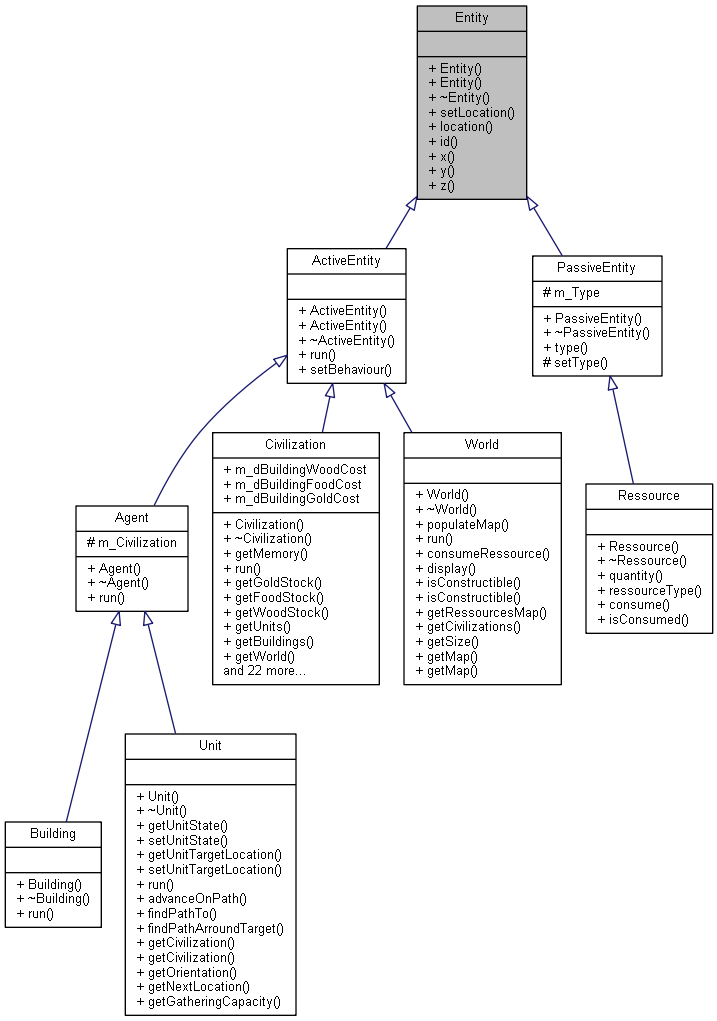


Figure 7 - Exemple de diagramme UML généré par Doxygen avec Graphviz/dot

# Conclusion

Doxygen est un outil très complet dont on découvre de nouvelles facettes à chaque nouvelle utilisation. Ce logiciel devient vite un outil indispensable à tout projet et brille par ses fonctionnalités et sa modularité. Cependant, la syntaxe utilisée pourra amener certains à ne pas apprécier son utilisation. Cette gêne peut facilement être occultée par l’installation de plugin de génération automatique de commentaires Doxygen à son environnement de développement préféré.