JavaDoc – Utiliser et Ecrire

Introduction

B. Mermet

Principes et intérêts

- Documentation intégrée au code et exportable dans différents formats
- L'intégration au code
  - diminue les risques d'obsolescence
  - Rend la documentation indissociable du code
- Intégré sous la forme d'un commentaire /** */
  - /* */ considéré comme un commentaire par les compilateurs et autres outils travaillant sur le code
  - /** distingué par l'outil javadoc

Utilisation

- Voir javadoc des API java standards

Documenter ? Quoi et où ?

- Quoi
  - (Une API)
  - (Un package)
  - Une classe / une interface
  - Une méthode / un constructeur
  - Un Attribut
- Où
  - juste avant la définition de l'élément à commenter

Structure d'un commentaire

- Allure générale :

```
/**
Une phrase.
Un texte qui peut être long. Ce texte peut être fait de plusieurs phrases.
Des champs structurés introduits par des étiquettes (tag)
*/
```

Ou :

```
/**
 * Une phrase.
 * Un texte qui peut être long. Ce texte peut être fait de plusieurs phrases.
 * Des champs structurés introduits par des étiquettes (tag)
 */
```

- Résumé :
  - La première phrase est une phrase de résumé.
  - Le texte est la description complète, informelle, en HTML.

Exemple de code documenté 🕮

```
/**
 * Cette classe permet de tester un peu la javadoc.
 * Il s'agit d'un exemple de classe toute simple. Cette classe est documentée. Elle contient
 * également un attribut documenté. De même, on commente certaines méthodes.
 * @author Bruno
 */
public class EssaiJavDoc {

    /** L'entier encapsulé par la classe. */
    public int x;

    /**
     * Accesseur en écriture pour x. Cette méthode prend tout simplement un entier...
     * et l'affecte à x comme on pouvait s'y attendre.
     * @param x l'entier que l'on veut stocker.
     */
    public void setX(int x) {this.x = x;}

    /**
     * Accesseur en lecture pour x.
     * @return la valeur encapsulée.
     */public int getX() {return x;}

}
```

Types d'étiquette

- Etiquettes "blocs"
  - @nomTag
  - Dans la partie "champs structurés"

- Etiquettes "en ligne"
  - {@nomTag}
  - Dans le corps du texte de commentaire

Liste non-exhaustive d'étiquettes

Commenter une classe

- @author nomAuteur
  - Une ligne par auteur

Commenter une méthode

- @exception classeException commentaire
- @throws classeException commentaire
    - Pour détailler une exception éventuellement levée
- {@inheritDoc}
    - Pour reprendre la description informelle de la super-classe
    - Ou pour reprendre la description d'un paramètre, retour ou exception de la super-classe
- @param nomParamètre description
    - Description d'un paramètre
- @return description
    - Description de la valeur retournée

Commenter un champ

- {@value}
  - Pour une constante, permet de faire référence dans le commentaire informel à sa valeur

Etiquettes diverses

- {@code *texte*} : pour considérer *texte* comme un extrait de code
- {@link package.classe#membre label} : lien vers un package, une classe ou une méthode avec "label" comme nom du lien. package.classe n'est pas nécessaire si la méthode ou l'attribut (membre) est dans la même classe.
- {@literal texte} : pour éviter l'interprétation de "texte" comme du html
- @see "chaine" / @see ... / @see package.classe#membre label
    - Ajout d'une section "see also"
- @since texte : spécifie la version (texte informel) depuis laquelle l'élément commenté a été introduit
- {@value package.class#membre} : pour reprendre la valeur d'une constante
- @version texte : pour spécifier le numéro de la version courante

Pour aller plus loin

- Commentaires des packages et api
- La commande javadoc
- Les doclets pour générer des documents dans des formats autres que HTML
- Les étiquettes non présentées ici
- Intégration dans Eclipse