javadoc

Université de Nice - Sophia Antipolis
Richard Grin
Version 1.6.1 – 22/11/18

Généralités

- javadoc produit de la documentation en partant de commentaires particuliers insérés dans le code source des classes (/** . . . */)
- On peut ainsi documenter
  - paquetages
  - classes ou interfaces
  - variables d'instance
  - méthodes
- Sauf pour les paquetages, les commentaires doivent être placés juste avant ce qu'ils commentent

Format des commentaires

- Les commentaires peuvent contenir
  - du texte simple
  - des tags HTML de mise en forme de texte (italique, caractère gras, caractères à espacement fixe,…) ; un tag bien utile est **<code>** (et **</code>**) pour inclure du code dans les commentaires
  - des tags spéciaux à javadoc, qui commencent par le caractère @ : **@author**, **@version**, **@param**,…

Format des commentaires

- Il est possible de passer à la ligne pour couper les lignes trop longues ; javadoc ignore les « * » placés en début de ligne

Caractères spéciaux

- Les caractères liés à HTML comme « < » ou « > » sont interprétés spécialement par javadoc
- « < » doit être écrit « **<** » dans la javadoc
- Depuis le JDK 5.0 le tag **@literal** permet d'insérer plus facilement les caractères spéciaux ; par exemple **{@literal A<B>C}** sera affiché « **A<B>C** » par la javadoc
- **{@code A<B>C}** fait la même chose en ajoutant la police de caractères du tag HTML **<code>**

Insérer du code

- Si le code à insérer fait plusieurs ligne il faut utiliser à la fois **{@code** et la balise HTML **<pre>** :

```
<pre>{@code
...
...
}</pre>
```

Insérer une référence

- Les tags **`@see`** et **`@link`** permettent d'insérer une référence vers une autre partie de la documentation ou même vers une autre documentation javadoc quelconque
- Un simple lien HTML convient pour insérer un lien HTML dans du texte :

```
Voir <a href="http://google.com">Google</a>
```


Insérer une référence

- **`@see`** ajoute une entrée dans la section « See also » de la documentation
- **`@link`** insère un lien vers une autre partie de la javadoc
- Plusieurs formats peuvent être utilisés pour indiquer ces références

Formats pour les références (1)

- chaîne de caractères quelconque : la chaîne sera affichée telle quelle dans la javadoc
  Exemple :

```
@see "The Java Programming Language"
```

- *label* : lien vers une adresse Web (adresse absolue ou relative)
  Exemple :

```
@see <a href="spec.html#section">Java
Spec</a>
```


Formats pour les références (2)

- *package.Classe#membre label* : lien vers un autre endroit de la javadoc (ou vers une autre javadoc si l'option **`-link`** est utilisée au lancement de la commande javadoc
  - Le texte du lien sera « label »
  - Remarquez le # à la place d'un « . » entre le nom de la classe et le nom de la méthode

Formats pour package.Classe#membre

- Le membre peut être un constructeur, une méthode ou une variable d'instance
- On peut aussi désigner une classe par package.Classe, ou une classe interne par package.Classe.ClasseInterne (ne jamais omettre la classe englobante, même si le commentaire est dans la classe englobante)
- On peut aussi désigner un paquetage :

```
@see fr.unice.truc
```


Formats pour package.Classe#membre

- Si la classe appartient au paquetage de la classe documentée ou si la classe est importée, on peut omettre le nom du paquetage :

```
@see Classe
```

- Si le membre appartient à la classe qui est documentée, on peut omettre package.Classe :

```
Utilisez la méthode {@link
#getComponentAt(int, int) getComponentAt}.
```

- Si la méthode n'est pas surchargée, le nom de la méthode suffit ; sinon il faut indiquer sa signature

Section « see also »

- @see crée une entrée dans la section « see also » de la documentation
- Exemples :
  - **`@see java.lang.Integer#parseInt`** ***`label`***
  - **`@see <a href="…">`***`label`***`</a>`**
    (si le 1er caractère est "<", c'est un lien HTML)
  - **`@see "`***`texte quelconque`***`"`**

Exemple de @see

- @see "The Java Programming Language"
- @see Java Spec
- Pour désigner des méthodes de la classe ou d'une autre classe :
  - @see equals
  - @see String#equals(Object) equals

Pour les méthodes surchargées

Le label qui sera affiché

Les classes sont cherchées dans le classpath

Liens entre commentaires

- {@link nom-classe#membre label}
  permet de placer un lien n'importe où dans la documentation (ne pas oublier les accolades)
- Exemple :
  Utilise la méthode {@link #getComponentAt(int, int) getComponentAt}.
- @linkplain a une syntaxe identique à @link mais le label est affiché dans la police de caractères du texte ordinaire et pas dans la police du code

Commentaires @since

- @since permet d'indiquer une version depuis laquelle ce qui est commenté a été introduit
- Exemple :
  @since JDK1.1

Commentaires de classe et d'interface

- @param <*E*> description de ce que représente le paramètre de type <E> (depuis le JDK 5.0)
- @author nom
  indique l'auteur (plusieurs fois si plusieurs auteurs)
- @author texte
  indique le ou les auteurs en utilisant le texte
- @version texte
  précise la version
- Ces 2 tags ne sont utilisés dans la documentation que si on donne les options -author et -version de la commande javadoc

Commentaires de méthodes

- Tous les tags de même type doivent se suivre
- Les descriptions peuvent s'étaler sur plusieurs lignes
- **`@param`** ***`paramètre`*** description
  documente un ***`paramètre`*** de la méthode
- **`@param <`***`T`***`>`** description
  documente un ***`paramètre`*** de type <*T*> de la méthode (ne pas omettre les < et >)
- **`@return`** description
  documente ce que retourne la méthode
- **`@throws`** ***`classe_exception`*** description
  documente une exception (on peut aussi utiliser @exception)

Résumé de commentaire

- La première phrase du commentaire d'un membre d'une classe constitue le résumé du commentaire
- Ce résumé est affiché dans la section « résumé »
- Le reste peut être vu en suivant le lien de la section résumé
- La « première phrase » se termine
  - par un point suivant d'un espace ou d'une fin de ligne
  - ou par un tag javadoc comme **`@param`** ou **`@return`**

Exemple de documentation de méthode

```
/**
 * Returns the character at the specified index. An index
 * ranges from <code>0</code> to <code>length() - 1</code>.
 *
 * @param   index  the index of the desired character.
 * @return    the desired character.
 * @throws StringIndexOutOfRangeException
 *             if the index is not in the range <code>0</code>
 *             to <code>length()-1</code>.
 * @see       java.lang.Character#charValue()
*/
public char charAt(int index) {
```


Héritage des commentaires

- Si la méthode d'une classe n'a pas de commentaire, elle hérite automatiquement des commentaires de la méthode qu'elle redéfinit ou implémente (s'ils existent)
- Avant la version 5 de java, si la méthode ajoutait un commentaire, il fallait ajouter « **`{@inheritDoc}`** » pour demander cet héritage des commentaires
- Depuis la version 5, s'ils manquent, les commentaires pour une méthode, un paramètre, la valeur retour ou une exception sont hérités automatiquement de la classe mère ou de l'interface implémenté ou hérité

Commentaires de paquetage

- Ils doivent être placés dans un fichier nommé **`package-info.java`** placé dans le répertoire des fichiers source du paquetage, avec les fichiers .java
- Avant le JDK 5.0, le fichier devait s'appeler **`package.html`** ; le JDK 5.0 accepte un des 2 fichiers (mais pas les 2)
- Les contenus de **`package-info.java`** et de **`package.html`** diffèrent

Exemple de `package-info.java`

```
/**
 * Ce paquetage ... .
 * <p>
 * ...
 * (voir la classe {@link truc.Classe1})
 * ...
 * @since 1.1
 * @see java.awt
 */
package truc.machin;
```


Commentaires généraux

- De même, un fichier de nom quelconque (typiquement un fichier placé à la racine des fichiers source et nommé **`overview.html`**) peut être placé dans le répertoire parent de tous les fichiers source ; ce fichier contient des commentaires sur tout le code
- Le contenu de ce fichier sera affiché quand l'utilisateur cliquera sur le lien « Overview » de la documentation si on génère la documentation avec l'option « **`-overview` *`nom-fichier`*`.html`** »

`package.html`

- Il est recommandé d'utiliser plutôt package-info.java
- C'est un fichier html ordinaire
- Tout ce qui est entre <BODY> et </BODY> se retrouve dans la documentation
- La 1ère phrase (jusqu'à un ".") doit être un résumé de ce que contient le paquetage
- Exemples de tags utilisables dans le corps du fichier html : @see, @since, @link

Exemple de `package.html`

```
<html>
<body>
Ce paquetage ….
. . .
@since 1.1
</body>
</html>
```


`package-info.java`

- Il ressemble à un commentaire javadoc habituel
- La 1ère phrase (jusqu'à un ".") doit être un résumé de ce que contient le paquetage
- Exemples de tags utilisables dans le corps du fichier html : @see, @since, @link

Fichiers annexes

- Si la documentation d'un paquetage utilise des fichiers annexes, par exemple des images, on doit les placer dans un répertoire nommé `doc-files` du paquetage
- Ces fichiers pourront être référencés par un nom relatif commençant par `doc-files` :

```
/**
 * Image du bouton :
 * <img src="doc-files/Bouton.gif">
 */
```


Générer une javadoc

Syntaxe de la commande

- **`javadoc`** [*options*…] [*cheminsSourcesClasses*…] [*nomsPaquetages*]
- Si on donne des noms de paquetages en paramètres, les fichiers sources doivent être dans un répertoire qui correspond au nom du paquetage
- L'option **`-sourcepath`** indique sous quel répertoire trouver les sources des classes des paquetages quand on donne *nomsPaquetages* ; elle n'est pas utilisée si on donne *cheminsSourcesClasses*
- *Ce cours ne donne qu'une petite partie de la syntaxe ; consultez la documentation officielle pour les compléments*

Options de la commande (1)

- Options (quelques options seulement) :
  - **`-d` *`répertoire`*** : indique le répertoire dans lequel mettre la documentation (répertoire courant par défaut)
  - **`-protected`**, **`-public`**, **`-package`**, **`-private`** : indique les membres et constructeurs qui apparaîtront dans la documentation (**`protected`** par défaut)
  - **`-author`**, **`-version`** : l'auteur et la version seront affichés (par défaut, ils ne le sont pas)

Options de la commande (2)

- **`-link URLautreJavadoc`** : permet de faire des liens vers une autre documentation *javadoc* préexistante (on peut donner plusieurs options –link) ; on peut donner un chemin relatif (tenir alors compte des chemins de *destination* de la javadoc) ou absolu
- **`-sourcepath`** : indique où trouver les fichiers sources des paquetages dont on veut créer la documentation (par défaut, ils sont cherchés dans le *classpath*)
- **`-classpath`** : comme pour les autres outils

Options de la commande (3)

- **`-use`** : génère un lien qui indique par qui est utilisé une classe ou un paquetage
- **`-overview` *`nom-fichier`*`.html`** : indique un fichier qui donne une vision d'ensemble du code du projet
- plusieurs options pour donner des titres, en-têtes ou pieds de pages à la documentation (voir documentation de Sun)

Exemples

```
javadoc -link http://www-
  mips.unice.fr/Java/jdk1.2/api
    fr.unice.toto.librairie
```


Option `-linkoffline`

- Il est possible que la javadoc d'un paquetage ne soit pas accessible pendant l'exécution de javadoc car javadoc a seulement besoin du fichier package-list placé avec la javadoc de l'autre paquetage
- En ce cas, on peut remplacer l'option **`-link`** par une option « **`-linkoffline` *`urlExterne repListeLocale`*** »
  - ***`urlExterne`*** est l'url qu'on aurait donné à l'option **`-link`**
  - ***`repListeLocale`*** est le répertoire où on a rangé le fichier package-list manquant, que l'on a réussi à récupérer par un moyen ou par un autre

Fichier `package-list`

- Le fichier **`package-list`** est généré automatiquement par javadoc ; il est placé dans le répertoire qui contient la javadoc générée par une commande javadoc unique
- Il contient les noms des paquetages dont le répertoire contient la javadoc
- Il est utilisé par javadoc pour faire les liens HTML qui référencent des éléments d'un autre paquetage : quand il y a une option **`-link`**, il utilise ce fichier pour savoir rapidement comment faire ces liens (il ne rentre pas dans les détails des fichiers associés à la javadoc extérieure)

Problème avec les liens

- Si on génère séparément la javadoc de 2 paquetages p1 et p2 inter-dépendants, le fichier **`package-list`** n'existera pas lors de la première exécution de javadoc et la javadoc de p1 n'aura pas de liens vers p2
- En ce cas, il faut lancer javadoc sur p1, puis sur p2, et encore une fois sur p1

javadoc avec NetBeans

- Générer la javadoc :
  Menu Run > Generate Javadoc
- Voir la javadoc :
  se placer sur la classe, la méthode ou une variable d'état et Alt + F1