

Comparación y discriminación de las etapas de Javadoc para java y Bluedoc para C++

De acuerdo con lo descrito en la Etapa 1, las fases de creación de documentación para Javadoc son:

1. Comprobar que el nombre del archivo de entrada, menos la extensión *.java*, corresponde a las reglas de sintaxis establecidas en los nombres de clases de Java.
2. El *path* relativo, especificado en el paquete al cual pertenece el archivo, es legal.
3. El enunciado en donde se declara el nombre del paquete corresponde con las reglas de sintaxis de declaración de paquetes.
4. Una vez que se han hecho estas pruebas, se comienza con el procesado del archivo.
5. Se realizan las acciones especificadas dentro de las opciones en línea de comandos.
6. Dentro de la estructura interna del archivo, verifica que todo cumpla con las reglas básicas de lenguaje.
7. Aunque se verifican algunas reglas, algunas otras se dejan pasar, para permitir la creación de la documentación aún antes de terminar la implementación.
8. Creación de los objetos de tipo *Doclet*, que realizan el trabajo de creación de los archivos HTML.
9. Finalmente, el objeto *Standard*, contenido en *com.sun.tools.doclets.standard* arregla y conjunta los HTML creados.

La documentación creada por Javadoc tiene las siguientes características:

- Añade al archivo fuente las convenciones de clases que fueron anuladas. (constructor vacío, declaración de métodos abstractos dentro de clases abstractas, etc).
- Añade declaraciones de retorno y de parámetros no documentadas.
- Trabaja únicamente con los *stubs* de las clases, no con las implementaciones de las mismas.
- Permite documentar un conjunto de archivos, o de manera simultánea, todas las clases que existen en un paquete y sus subpaquetes.
- Existen *links* que llevan a la descripción de todas las clases, para una navegación más dinámica, sin embargo, esto lo realiza en tiempo de compilación, no admite resultados incrementales.
- Se generan *warnings* en caso de que no se cumplan las reglas de Javadoc, como el uso de *tags* en lugares que no les corresponden (un ejemplo es *@param* en la documentación de una clase).

Asimismo, la invocación de Javadoc tiene la siguiente estructura, con las opciones especificadas en el anexo 1.1.

javadoc [opciones] [paquetes] [archivos fuente] [subpaquetes] [argFiles]

Para proponer una estructura para Bluedoc, es conveniente seguir de manera inversa la explicación anterior.

bluedoc [opciones] [paquetes] [archivos fuente]

Las opciones que aceptará Bluedoc serán las especificadas en el anexo 2.1, con las siguientes excepciones, que no serán incluidas:

-1.1

No existe una versión anterior, así que no es necesaria.

-author

La inclusión del nombre del autor será automática.

-bootclasspath *classpathList*

La necesidad de cambiar las clases *Core* del lenguaje es muy remota, por lo que no se considera necesario.

-charset

No es necesario especificar el conjunto de caracteres.

-docencoding *name*

Misma razón que en `-charset`.

-docfilessubdirs

La copia de los archivos corre a cargo del usuario, o utilizar la opción `-d`.

-encoding *name*

Misma razón que en `-charset`.

-extdirs *dirlist*

Especifica el lugar en donde residen las clases de Extensión (*i??*)

-J

No existe tal cosa como una Máquina Virtual en C++.

-linkoffline *extDocument*.

Se puede utilizar de manera genérica el modificador `-link`.

-locale

No existe la clase `java.util.Locale` en C++.

-nodeprecated

Es necesario especificar que una clase ha sido reemplazada.

-nodeprecatedlist

Misma razón de `-nodeprecated`.

-noqualifier *all / package1:package2*

No existen las rutas completamente calificadas.

-source

No existen las invariantes de Javadoc 1.4 en C++.

-subpackages *package1:package2*

No existen los subpaquetes.

-tag

No es bueno para la compatibilidad crear nuevos tags no documentados.

-use

El documento de `-use` no es tan utilizado por los usuarios de javadoc y API.

-version

Se incluirá la versión de manera automática.

El modificador de la línea de parámetros `-subpackages` será suprimido, puesto que ni siquiera existen los paquetes en C++. Sin embargo, la opción de `-packages` si será incluida, tal vez suene contradictorio pero tiene una explicación.

De manera interna, el conjunto de paquetes de Java tiene que ver con la jerarquía de clases, Herencia y de Implementación que exige un lenguaje 100% orientado a Objetos. Por otra parte, aunque C++ también es un lenguaje Orientado a Objetos, no lo es en su totalidad, por lo que pueden existir clases aisladas. Sin embargo, conjuntar estas clases en paquetes (por lo menos por parte de la documentación) serviría para clarificar, agrupar y distinguir distintos propósitos de clases creadas por usuarios.

El propósito de incluir una agrupación por paquetes es simplemente el de clarificar la documentación. De esta manera, la documentación SI incluye la opción para manejar paquetes.

También es interesante la opción de recibir un archivo de texto con todos los parámetros necesarios, pues en caso de que sea una lista extendida, será difícil su mantenimiento si es mandatario que el usuario los escriba todas las veces. Esta opción fue eliminada por considerar que este trabajo puede ser realizado de igual manera por un archivo `.bat`.

De esta manera, los únicos parámetros que manejará Bluedoc son los de opciones, paquetes y archivos fuente.

Las características de Bluedoc, basadas en Javadoc serán las siguientes:

- Añade al archivo fuente las convenciones de clases que fueron anuladas. (constructor vacío, declaración de métodos abstractos dentro de clases virtuales, etc).
- Añade declaraciones de retorno y de parámetros no documentadas.
- Trabaja únicamente con archivos de encabezados `.h`, no con las implementaciones de las mismas ni con los que contienen el método `main()`, en general, solo con archivos que contengan clases válidas.
- Permite documentar un conjunto de archivos, o de manera simultánea, todas las clases que existen en un paquete.
- Existen *links* que llevan a la descripción de todas las clases, para una navegación más dinámica, sin embargo, esto lo realiza en tiempo de compilación, no admite resultados incrementales.
- Se generan *warnings* en caso de que no se cumplan las reglas de Bluedoc, como el uso de *tags* en lugares que no les corresponden (un ejemplo es *@param* en la documentación de una clase).

Los Tags que manejará Bluedoc son los que se encuentran en el Anexo 2.2, a excepción de los siguientes, que no estarán incluidos:

{@docRoot}

El path relativo se calculará en base al archivo `index.html`, sin cambios.

{@inheritDoc}

Si la clase existe dentro del mismo paquete, se creará un link hacia ella, sin necesidad de copiar su contenido.

{@linkplain packageclass#memberclass}

Similar a @link

@see

Similar a @link.

@serial

No tengo documentación sobre serializable en C++, pero podría implementarse en el futuro.

@serialData *DataDescription*

Similar a @serial.

Finalmente, la primera aproximación a los pasos para la creación de los archivos HTML incluye los siguientes pasos:

1. Comprobar que el nombre del archivo de entrada, menos la extensión *.h*, corresponde a las reglas de sintaxis establecidas en los nombres de clases de Java (para seguir las mismas convenciones).
2. El enunciado en donde se declara el nombre del paquete corresponde con las reglas de sintaxis de declaración de paquetes.
3. Una vez que se han hecho estas pruebas, se comienza con el procesado del archivo.
4. Se realizan las acciones especificadas dentro de las opciones en línea de comandos.
5. Dentro de la estructura interna del archivo, verifica que todo cumpla con las reglas básicas de lenguaje.
6. Aunque se verifican algunas reglas, algunas otras se dejan pasar, para permitir la creación de la documentación aún antes de terminar la implementación.
7. Creación de los objetos de tipo *Doclet*, que realizan el trabajo de creación de los archivos HTML.
8. Finalmente, el objeto ***Standard***, contenido en las librerías de Bluedoc arregla y conjunta los HTML creados.

Con la definición de las etapas de la creación de los archivos de documentación, concluye la 2 etapa del proyecto.

Anexo 2.1 Opciones modificadoras de Bluedoc

-bottom *text*

Especifica un texto que será incluido en la parte de debajo de cada archivo obtenido.

-breakiterator

Cambia la forma de obtener la primer oración de cada bloque de comentarios, misma que se utilizará en el resumen.

-classpath *classpathlist*

Especifica la localización de las clases referenciadas por los archivos fuente.

-d *directory*

Especifica el directorio en donde se colocará los archivos resultantes.

-doclet *class*

Especifica el archivo class que contiene el *doclet* para la generación de los documentos. Este archivo debe de contener el método `start()` (método raíz).

-docletpath

Especifica el lugar en donde se encuentra el archivo class que contiene el *doclet*.

-doctitle *title*

Especifica un texto que servirá como título en la parte superior central del archivo `overview-summary.html`.

-exclude *package1:package2*

Especifica los paquetes que no se tomarán en cuenta en el proceso de documentación.

-footer *footer*

Especifica un texto que será colocado en la parte inferior de cada archivo generado, a la derecha de la barra de navegación más baja.

-group *groupheading package1:package2*

Separa todos los paquetes en grupos distintivos dentro del archivo `overview.html`

-header *header*

Especifica un encabezado que será colocado en todos los archivos generados, que estará a la derecha de la barra de navegación superior.

-help

Despliega la ayuda en línea de Javadoc, que es esta especificación de los parámetros posibles.

-helpfile *path/filename*

Especifica un archivo de ayuda distinto al generado por defecto, que es al que apunta el link HELP en la parte superior de los archivos html.

-link *extDocument*

Especifica un archivo html externo que contiene clases ya documentadas y que se quiere añadir a la nueva corrida de Javadoc que se quiere crear.

-linksource

Crea un archivo HTML para cada archivo fuente y los junta dentro de la documentación estandar, mostrando todos sus métodos, atributos y clases, sin importar si son privados o protegidos.

-nocomment

Elimina todos los cuerpos de comentarios, generando solamente un archivo con los nombres de métodos, clases y atributos.

-nohelp

Elimina el link al archivo de ayuda dentro de la barra de navegación superior.

-noindex

Omite la generación de un archivo index.html

-nonavbar

Previene la generación de la barra de navegación, tanto superior como inferior.

-nosince

Omite el texto generado por el *tag* @since.

-notree

Omite la jerarquía de clases o interfaces en los archivos de documentación generados.

-overview path

Especifica que archivo debe ser utilizado como overview.html en lugar de generar uno nuevo.

-package

Muestra solo los paquetes y las clases protegidas, publicas y miembros.

-private

Muestra todas las clases y miembros.

-protected

Muestra solo las clases y miembros públicos y protegidos. Es la opción por defecto.

-public

Muestra solo las clases y miembros públicos.

-quiet

Elimina todos los mensajes que no sean de error o de advertencia.

-serialwarn

Genera advertencias en tiempo de compilación para mostrar *tags* que no hayan sido escritos pero que sean necesarios. Un ejemplo es un método con 3 parámetros y solo 2 @param.

-sourcepath sourcepath

Especifica el lugar en donde se encuentran los archivos fuentes a procesar.

-splitindex

Divide el index en diferentes archivos, alfabéticamente, un archivo por letra.

-stylesheetfile path

Especifica un archivo .ccs distinto al generado por defecto.

-taglet class

Especifica la clase que contiene los *taglets* utilizados en la generación de la documentación.

-tagletpath

Especifica en donde se encuentra la clase que contiene los *taglets*.

-verbose

Provee mensajes detallados del proceso que se ejecuta al correr Javadoc.

-windowtitle title

Especifica el texto que será incluido como Título de la pagina de documentación, es decir, dentro de los tags de HTML <title></title>

Anexo 2.2 Tags disponibles de Bluedoc.

@author

Utilizado para especificar el autor de dicho archive.

@deprecated

Especifica que el API incluido no deberá de utilizarse en las nuevas versiones, aunque puede funcionar, pero que no es recomendado.

@exception

Sinónimo del *tag* @throws.

{@link packageclass#memberclass}

Crea un link a la clase determinada como argumento.

@param parameter-name description

Añade el parámetro a la sección de parámetros del método.

@package name

Indica que la clase pertenece al paquete especificado.

@return description

Añade el objeto retorno a la sección de Retorno. Únicamente válido en métodos.

@since

Añade una especificación aclarando a partir de cual versión se utiliza el objeto documentado.

@throws

Añade un comentario a la sección de excepciones y *throws*.

{@value}

Despliega el valor de todas las constantes declaradas.

@version

Añade la versión del archivo que se esta documentando.