Tema 4 : Herramientas de generación de documentación

Ciclo Superior DAW

Asignatura: Despliegue de aplicaciones web

Curso 20/21

Introducción

- En este capítulo veremos aspectos como:
  - Herramientas de generación automática de documentación.
  - Introducción a Javadoc.
  - Patrones de código.

Generación de documentación

- Un generador de documentación es una **herramienta de programación que genera documentación** destinada a los programadores (documentación de API), a usuarios finales, o a ambos, **a partir de un conjunto de código fuente especialmente documentado**, y en algunos casos, archivos binarios.

Generación de documentación

- Consiste en **incluir en el código fuente un tipo especial de comentarios** con información de cada elemento del código: clase, función, etc.
- Una vez finalizado, la herramienta analiza el código y genera la documentación

Generación de documentación

- En el siguiente enlace se pueden ver una comparativa de los sistemas de generación de documentación más salientables

  https://en.wikipedia.org/wiki/Comparison_of_documentation_generators

Actividad 1

Comparamos las características de varios generadores de documentación

Javadoc

- Javadoc es un generador de documentación de Java.
- Los **comentarios de documentación comienzan con /** y finalizan con */** y pueden aparecer justo antes de una clase, campo (variable de clase u objeto), constructor o definición de método.

Etiquetas Javadoc

- Las etiquetas existentes para documentar el código fuente son:
  - **@author**: Autor de la clase. Sólo para las clases.
  - **@version**: Versión de la clase. Solo para clases.
  - **@see**: Referencia a otra clase, ya sea del API, del mismo proyecto o de otro.
  - **@param**: Descripción parámetro. Una etiqueta por cada parámetro.

Etiquetas Javadoc

- Las etiquetas existentes para documentar el código fuente son:
  - **@return**: Descripción de lo que devuelve. Sólo si no es void. Podrá describir valores de retorno especiales según las condiciones que se den, dependiendo del tipo de dato
  - **@throws**: Descripción de la excepción que puede propagar. Habrá una etiqueta throws por cada tipo de excepción.
  - **@deprecated**: Marca el método como obsoleto. Solo se mantiene por compatibilidad.
  - **@since**: Indica el nº de versión desde la que existe el método.

Ejemplos prácticos

- Documentación de una clase

```
/**
 * @ author      Nombre y apellidos del autor
 * @ version     1.6 (versión actual del programa)
 * @ since       1.2 (versión del paquete a partir de la cual se añadió la clase)
 */
public class Prueba {
    // cuerpo de la clase
}
```

Ejemplos prácticos

- Documentación de un método

```
/**
 * Descripción corta de una sola línea.
 * < p>
 * Descripción larga, de varias líneas,
 * de existir, iría aquí
 * < p>
 * Y más explicaciones, si hiciera falta
 * en líneas sucesivas separadas por
 * párrafos HTML
 * @ param variable Descripción del parámetro
 * @ return Descripción del valor devuelto.
 */
public  int nombreMetodo (...) {
    // Cuerpo del método
}
```

Ejemplos prácticos

- Documentación de variables

```
/**
 * Descripción de la variable x.
 */
public  int x;
```

Ejemplos prácticos

- Documentación de paquetes
    - **La descripción de un paquete va en un archivo aparte llamado package-info.java** que debe encontrarse en el directorio del paquete y que contiene la declaración del paquete, precedido inmediatamente por la descripción del mismo.

Ejemplos prácticos

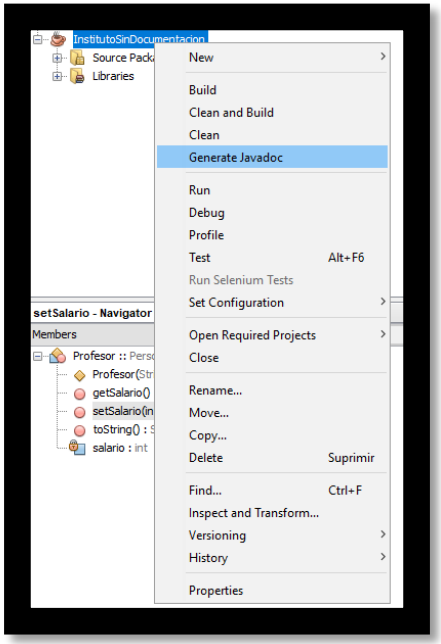
- Documentación de paquetes
    - **La descripción de un paquete va en un archivo aparte llamado package-info.java** que debe encontrarse en el directorio del paquete y que contiene la declaración del paquete, precedido inmediatamente por la descripción del mismo.

Actividad 2

Documentar una clase, un método y una variable del ejemplo adjunto

Generación de documentación en Netbeans

- Debemos hacer clic en el proyecto→ Generate Javadoc…



Generación de documentación en Netbeans

- Esto nos generará una serie de páginas HTML en el fichero dist. Podremos navegar por las diferentes páginas visualizando toda la información.

Actividad 3

Generar la documentación de la aplicación previamente documentada

Patrones de código

- Son **pequeños trozos de código reutilizable que contienen ciertas partes variables que se llenarán dependiendo del contexto.**

- Su uso **permite no tener que escribir un código relativamente repetitivo** (como puede ser un bucle) o no tener que recordar la sintaxis de una pieza de código más o menos compleja.

Patrones de código

- Cada patrón tiene un nombre, para acceder a ellos debemos escribir ese nombre y si estamos utilizando un IDE que tenga autocompletado nos va a mostrar todas las opciones

Patrones de código

- Por ejemplo, en Netbeans, si escribimos **for** y escogemos iterar sobre matriz, obtendremos:

```
for ( int i = 0; i <  array.length; i++) {
}
```

Actividad 4

Utilizando el IDE que queramos, probaremos el uso de algunos de los patrones de códigos existentes

Tema 4 : Herramientas de generación de documentación

Ciclo Superior DAW

Asignatura: Despliegue de aplicaciones web

Curso 20/21