Menulis Java Documentation

Javadoc?

javadoc itu merupakan dokumentasi dari class-class java, dokumentasi nya berisikan seluruh class, interface, dan metode-metode yang terdapat di dalamnya

Mengapa membuat javadoc?

Agar kita & orang lain tahu apa maksud & tujuan kode yang ditulis & mempermudah penggunaan class bagi pihak lain yang mau menggunakan atau juga sebagai dokumentasi pribadi untuk menghindari kita lupa fungsi dari class-class atau method yang telah kita buat

Dimana menulis javadoc?

biasanya javadoc itu berada sebelum deklarasi class, metode atau property, contoh :

```
/**
 *  BACA DENGAN TELITI !!!
 *
 *  Anda boleh menggunakan, mengubah, menghapus, menambah, dan
 *  melakukan hal yang anda inginkan ke dalam source code ini.
 *
 *  Dan saya tidak bertanggung jawab atas kesalahan,
 *  bug, atau keanehan yang ada dalam source code ini, dan
 *  saya juga TIDAK MENERIMA PERTANYAAN dari Anda
 *  mengenai isi source code ini.
 *
 *  Semoga source code ini bermanfaat bagi Anda, dan juga
 *  bermanfaat bagi project yang sedang Anda buat.
 *
 *  Salam saya, Agung Pramono
 *
 *  Dibuat Tanggal : 2019 Sept 8
 */
package com.agung.bean;

import java.io.Serializable;

/**
 * Alamat merupakan representasi dari Alamat orang class ini digunakan oleh
 * class Orang sebagai Alamat tempat tinggal orang yang bersangkutan
```

```
 *
 * @author agung
 */
public class Alamat implements Serializable {

    /**
     * jalan alamat
     */
    private String jalan;
    /**
     * kodepos alamat
     */
    private String kodePos;
    /**
     * kota alamat
     */
    private String kota;
    /**
     * negara alamat
     */
    private String negara;

    /**
     * membaut Alamat baru tanpa parameter
     */
    public Alamat() {
        // TODO Auto-generated constructor stub
    }

    /**
     * mendapatkan jalan alamat
     * @return jalan
     */
    public String getJalan() {
        return jalan;
    }

    /**
     * mengubah nilai jalan alamat
     * @param jalan
     *              jalan yang baru
     */
    public void setJalan(String jalan) {
        this.jalan = jalan;
    }
```

```
    /**
     * mendapatkan kodepos alamat
     * @return kodePos
     */
    public String getKodePos() {
        return kodePos;
    }

    /**
     * mengubah nilai kodepos alamat
     * @param kodePos
     *              kodepos yang baru
     */
    public void setKodePos(String kodePos) {
        this.kodePos = kodePos;
    }

    /**
     * mendapatkan kota alamat
     * @return the kota
     */
    public String getKota() {
        return kota;
    }

    /**
     * mengubah nilai kota alamat
     * @param kota
     *              kota yang baru
     */
    public void setKota(String kota) {
        this.kota = kota;
    }
}
```

Tag-tag Javadoc

Tag	Fungsi
`@author`	Untuk mencantumkan nama penulis kode biasanya ditulis di leve class
`@parameter`	Mendefinisikan parameter yg akan dilewatkan dalam sebuah method
`@return`	Mendefinisikan nilai kembalian dari sebuah method
`@see`	Menampilkan “See Also” dan juga untuk merefer ke kelas yang bersangkutan
`@version`	Menampilkan versi dari kode yang ditulis

Tag	Fungsi
`@throws`	Mendefinisikan exception yg akan ditimbulkan oleh sebuah method. Perhatikan bahwa kode Anda harus menunjukkan exception yang dilemparkan agar tag ini dapat divalidasi. Kalau tidak, Javadoc akan menghasilkan kesalahan. `@exception` adalah tag alternatif.
`@Override`	Melakukan pemeriksaan untuk melihat apakah metode ini mengoverride. biasanya digunakan untuk kelas abstract dan interface.
`@since`	Versi sejak fitur ditambahkan.
`{@link}`	Digunakan untuk membuat tautan ke kelas atau metode lain. Contoh: {@link Foo # bar} tautan ke bilah metode milik kelas Foo. Untuk menautkan ke metode di kelas yang sama, cukup sertakan #bar.
`@deprecated`	Memungkinkan pengguna tahu kelas atau metode tidak lagi digunakan. Tag ini akan diposisikan secara langsung di Javadoc. Biasanya digunakan secara bersamaan dengan tag `@see` atau `{@link}`.

Comment vs Javadoc

Sebuah comment biasanya ditulis sbb:

```
// sample comment...

/*
sample comment
*/
```

javadoc tidak akan memproses komen seperti diatas. Agar dapat diproses maka komen harus ditulis sbb:

```
/**
*
* comment javadoc
*
*
*/
```

Dimana Javadoc ditulis ?

Tag javadoc dapata diletakan sebelum kelas atau metode (tidak perlu ruang antara deskripsi, kelas atau metode).

Elemen apa yang bisa dipasang tag Javadoc ?

Kita dapat menambahkan tag Javadoc ke kelas, metode, dan interface. * Untuk tag `@author` dan `@version`, tambahkan hanya ke kelas dan interface. * Tag

`@param` hanya bisa ditambahkan ke metode dan konstruktor. * Tag `@return` hanya ditambahkan ke metode.

Access Modifier publik vs private di Javadoc

Javadoc hanya mencakup kelas, metode, dll yang ditandai sebagai publik. Elemen private tidak termasuk. Standarnya adalah bahwa kelas atau metode hanya tersedia untuk paket. Dalam hal ini, itu tidak termasuk dalam Javadoc.

Urutan Penggunaan tag javadoc

Oracle menyarankan dalam menggunakan tag sesuai dg urutan berikut:

```
@author (classes and interfaces)
@version (classes and interfaces)
@param (methods and constructors)
@return (methods)
@throws (@exception is an older synonym)
@see
@since
@serial
@deprecated
```

Tag `@param`

ag @param hanya berlaku untuk metode dan konstruktor, yang keduanya mempunyai parameter. Setelah tag `@param`, tambahkan nama parameter, dan kemudian deskripsi parameter, dalam huruf kecil, tanpa periode, seperti ini:

```
/**
*
* @param message pesan yang akan dikirim ke server
*/
public void sendMessage(String message){

}
```

Deskripsi parameter adalah frasa, bukan kalimat lengkap.

Tag `@return`

Digunakan untuk metode yang mengembalikan nilai. Jika suatu metode mengembalikan `void` maka hindari memakai tag `@return` untuk menghindari kesalahan saat mengkompilasi Javadoc.

Tag `@throws`

Tag `@throws` ke metode hanya jika metode melempar jenis kesalahan tertentu. Contoh :

```
/**
*
* @trhows IOException jika format input salah
*/
public void readFile()throws IOException {
}
```

Tag `@see`

Tag `@see` memberikan referensi lihat juga

```
    @see #field
    @see #Constructor(Type, Type...)
    @see #Constructor(Type id, Type id...)
    @see #method(Type, Type,...)
    @see #method(Type id, Type, id...)
    @see Class
    @see Class#field
    @see Class#Constructor(Type, Type...)
    @see Class#Constructor(Type id, Type id)
    @see Class#method(Type, Type,...)
    @see Class#method(Type id, Type id,...)
    @see package.Class
    @see package.Class#field
    @see package.Class#Constructor(Type, Type...)
    @see package.Class#Constructor(Type id, Type id)
    @see package.Class#method(Type, Type,...)
    @see package.Class#method(Type id, Type, id)
```

Tautan

Anda dapat membuat tautan ke kelas dan metode lain menggunakan tag `{@link}`. Berikut ini contoh dari standar kode Javadoc tentang membuat tautan:

```
/**
* First paragraph.
* <p>
* Link to a class named 'Foo': {@link Foo}.
* Link to a method 'bar' on a class named 'Foo': {@link Foo#bar}.
* Link to a method 'baz' on this class: {@link #baz}.
```

```
* Link specifying text of the hyperlink after a space: {@link Foo the Foo class}.
* Link to a method handling method overload {@link Foo#bar(String,int)}.
*/
public...
```

Untuk menautkan ke metode lain dalam kelas yang sama, gunakan format ini: `{@link #baz}`. Untuk menautkan ke metode di kelas lain, gunakan format ini: `{@link Foo # baz}`. Jika ingin menghindari syntaks hyperlink lebih baik menggunakan tag `<code>`. Jika ingin membuat referensi "lihat juga/see also", gunakan format ini: `@see #baz`. Untuk mengubah teks yang tertaut, letakkan kata setelah #baz seperti ini: `@see` metode `#baz` Baz.

Code Snippet di javadoc

Javadoc mendukung tiga fitur berbeda untuk markup kode. Antara lain tag HTML `<pre>` dan `<code>` dan tag Javadoc `{@code}`.

<pre>

contoh :

```
/**
 * <pre>
 * public class JavadocTest {
 *    // indentation and line breaks are kept
 *
 *    @SuppressWarnings
 *    public List<String> generics(){
 *      // '@', '<' and '>'  have to be escaped with HTML codes
 *      // when used in annotations or generics
 *    }
 * }
 * </pre>
 */
public class PreTest {}
```

output:

```
public class JavadocTest {
   // indentation and line breaks are kept
   @SuppressWarnings
   public List<String> generics(){
     // '@', '<' and '>'  have to be escaped with HTML codes
     // when used in annotations or generics
   }
 }
```

<code>

Dalam tag <code>, identasi akan diabaikan dan karakter khusus akan tetap ditampilkan.

contoh :

```
/**
 * Using <code>, indentation and line breaks are lost.
 * '@', '<' and '>'  have to be escaped with HTML codes.
 *
 * An annotation <code>@Foo</code>; and a generic List<String>.
 */
public class CodeHtmlTagTest {}
```

output :

```
Using <code>, indentation and line breaks are lost. '@', '<' and '>' have to be escaped with
```

{@code}

{@code} adalah tag Javadoc yang ada pada Java 5. Kutipan kode yang berada dalam {@code} akan menampilkan karakter khusus dengan benar sehingga tidak perlu diformat secara secara manual. Tapi, indentasi dan line break akan hilang. Ini bisa diperbaiki dengan menggunakan {@code} bersama dengan <pre>

contoh :

```
/**
 * Using {@code @code} alone, indentation will be lost, but you don't have to
 * escape special characters:
 *
 * {@code An annotation <code>@Foo</code>; and a generic List<String>}.
 */
public class CodeJavadocTagTest {}
```

output :

```
Using @code alone, indentation will be lost, but you don't have to escape special characters
```

<pre> + {@code}

Menggabungkan <pre> dan {@code}, indentasi dan jeda baris disimpan dan < dan > akan ditampilkan dengan benar. Namun, karakter @ sekarang dievaluasi sebagai tag Javadoc jika diikuti dengan karakter non-spasi. Yang lebih buruk: itu bahkan tidak bisa diformat dengan menggunakan kode angka HTML, karena kode angka HTML akan diklasifikasi oleh {@code}

```
/**
 * <pre>{@code
 * public class JavadocTest {
 *   // indentation and line breaks are kept
 *
 *  @literal @SuppressWarnings
 *   public List<String> generics(){
 *     // '<' and '>'  are displayed correctly
 *     // '@' CANNOT be escaped with HTML code, though!
 *   }
 * }
 * }</pre>
 */
public class PreTest {}
```

Output:

```
public class JavadocTest {
    // indentation and line breaks are kept
    &#64;SuppressWarnings
    public List<String> generics(){
        // '<' and '>'  are displayed correctly
        // '@' CANNOT be escaped with HTML code, though!
    }
}
```

Kesimpulan

Tabel berikut merangkum berbagai fitur markup kode Javadoc.

	`<pre>...</pre>`	`<code>...</code>`	`{@code ...}`
keep indentation & line breaks	yes	no	no
display '<' and '>' correctly	no	no	yes
display '@' correctly	no	no	yes
escape special characters via HTML number codes	yes	yes	no need to esca

Pakai yang mana ?

Melihat tabel di atas, sayangnya, tidak ada pilihan terbaik. Opsi mana yang digunakan tergantung pada konten potongan kode yang ingin Anda sematkan di Javadoc Anda. Panduan berikut dapat diturunkan untuk situasi yang berbeda:

Situation	Code Markup Feature	Rationale
inline code snippet	`{@code ...}`	With `{@code...}` you don't need to escape special characters. For inline snippets, it doesn't matter that line breaks are lost
multi-line Java code snippet	`<pre>...</pre>`	For multi-line snippets, you need line breaks. So only `<pre>` and the combination of `<pre>`and `{@code...}`. However, only `<pre>` allows the use of `@` (escaped using HTML number codes), which you need for Java code containing annotations.
mutli-line HTML / XML code snippet	`<pre>{@code...}</pre>`	In HTML or XML code you probably need a lot of `<` and `>` while `@` is not as important, so it doesn't matter that `@` cannot be displayed. If you need an `@`, you have to fall back on `<pre>`

Langkah-langkah menulis java doc

1. Tulis source code
2. Tambahkan plugin jika menggunakan maven

```
<build>
```

```
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-javadoc-plugin</artifactId>
                <version>3.0.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
                <tags>
                ...
                </tags>
            </plugin>
        </plugins>
    </build>
```

3.Jalankan perintah `mvn javadoc:javadoc`

Referensi

1. https://reflectoring.io/howto-format-code-snippets-in-javadoc/
2. https://dzone.com/articles/a-guide-to-formatting-code-snippets-in-javadoc
3. https://alvinalexander.com/java/edu/pj/pj010014
4. https://www.tutorialspoint.com/java/java_documentation.htm
5. https://www.oracle.com/technetwork/articles/java/index-137868.html