

Mengenal Annotations pada Pemrograman Java

Amru Rosyada

taka86@gmail.com

<http://amrurosyada.blogspot.com>

Lisensi Dokumen:

Copyright © 2003-2008 IlmuKomputer.Com

Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

Java merupakan bahasa pemrograman yang sudah sangat dikenal banyak kalangan, mulai dari pelajar, akademisi sampai game developer menggunakannya sebagai core dalam pembuatan program.

Kali ini kita akan membahas apa itu, anotation dan apa kegunaan dari annotation.

Pendahuluan

Annotation merupakan informasi data tentang kode program tetapi tidak akan berdampak secara langsung pada kode, atau lebih dikenal dengan nama meta tag.

Lalu apa kegunaan dari annotation ?

1. Memberikan informasi kepada compiler - Annotation dapat digunakan oleh compiler untuk mendeteksi error atau suppress warning.
2. Compiler-time dan deployment-time processing - Software tools dapat memproses informasi dari annotation untuk men-generate code, XML file, dan masih banyak lagi.
3. Runtime processing - Beberapa annotation akan diperiksa pada waktu runtime.

Annotation dapat diaplikasikan pada *class*, *field*, *method*, dan *elemen* program yang lain.

Isi

Annotation sering berdiri sendiri, dan mempunyai elemen baik dengan nama ataupun tidak :

```
@Author(  
    name = "Benjamin Franklin",  
    date = "3/27/2003"  
)  
class MyClass() { }
```

atau

```
@SuppressWarnings(value="unchecked")  
void myMethod() { }
```

Jika hanya ada satu elemen saja bernama "value," maka nama dapat dihilangkan, sbb :

```
@SuppressWarnings("unchecked")  
void myMethod() { }
```

Begitu juga jika annotation tidak mempunyai elemen, tanda kurung dapat dihilangkan, sbb :

```
@Override  
void myMethod() { }
```

Documentation

Annotation sering digunakan untuk menggantikan komentar pada program.

Pada program biasanya menambahkan komentar pada program untuk memberikan informasi2 yang penting, sbb :

```
public class Generation3List extends Generation2List {  
  
    // Author: John Doe  
    // Date: 3/17/2002  
    // Current revision: 6  
    // Last modified: 4/12/2004  
    // By: Jane Doe  
    // Reviewers: Alice, Bill, Cindy  
  
    // class code goes here  
  
}
```

Untuk menambahkan dengan annotation seperti pada komentar pada class Generation3List diatas, kita harus mendefinisikan tipe annotation. Sintaknya sebagai berikut :

```
@interface ClassPreamble {  
    String author();  
    String date();  
    int currentRevision() default 1;  
    String lastModified() default "N/A";  
    String lastModifiedBy() default "N/A";  
    String[] reviewers(); // Note use of array  
  
}
```

jika dilihat sekilas pendefinisian tipe annotation mirip dengan pendefinisian interface, bedanya pada tipe annotation didahului dengan karakter @ (@ = "AT" sebagai tipe Annotation) dan bisa diberi nilai default.

Kemudian kita bisa mendefinisikan tipe annotation pada class Generation3List, sbb :

```
@ClassPreamble (  
    author = "John Doe",  
    date = "3/17/2002",  
    currentRevision = 6,  
    lastModified = "4/12/2004",  
    lastModifiedBy = "Jane Doe"  
    reviewers = {"Alice", "Bob", "Cindy"} // Note array notation  
)  
public class Generation3List extends Generation2List {
```

```
// class code goes here
```

```
}
```

Catatan : Untuk membuat informasi2 pada ClassPreamble muncul pada saat pembuatan javadoc, kita harus mendefinisikan anotasi `@Documented`, sbb:

```
import java.lang.annotation.*; // import this to use @Documented
```

```
@Documented
@interface ClassPreamble {

    // Annotation element definitions

}
```

Annotation digunakan oleh compiler

Ada tiga tipe annotation yang secara otomatis akan didefinisikan : `@Deprecated`, `@Override`, dan `@SuppressWarnings`.

@Deprecated -- Digunakan untuk menandai bahwa elemen yang ditandai sudah tidak digunakan lagi, mungkin karena suatu alasan dari developer. Kita menggunakan tag `@deprecated` untuk mendokumentasikannya, sbb :

```
// Javadoc comment follows
/**
 * @deprecated
 * explanation of why it was deprecated
 */
@Deprecated
static void deprecatedMethod() { }
```

@Override -- Digunakan untuk menandai bahwa method merupakan method yang di-override dari superclass atau interface.

```
// mark method as a superclass method
// that has been overridden
@Override
int overriddenMethod() { }
```

compiler akan menampilkan kesalahan jika terjadi kesalahan pada waktu melakukan overridden.

@SuppressWarnings -- Digunakan untuk tidak memunculkan pesan peringatan, misalnya kita menggunakan method yang telah deprecated, kita bisa menggunakan annotation ini untuk menghilangkan pesan peringatan bahwa method yang kita pakai adalah deprecated :

```
// use a deprecated method and tell
// compiler not to generate a warning
@SuppressWarnings("deprecation")
void useDeprecatedMethod() {
    objectOne.deprecatedMethod(); //deprecation warning - suppressed
}
```

agar annotation bisa dilihat saat runtime, maka gunakan
@Retention(RetentionPolicy.RUNTIME), sbb:

```
import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@interface AnnotationForRuntime {

    // Elements that give information
    // for runtime processing

}
```

Penutup

Diharapkan dengan adanya artikel ini bisa membantu dalam meningkatkan kemajuan teknologi informasi di Indonesia dan mendukung suksesnya IGOS

Referensi

[Http://java.sun.com](http://java.sun.com)

Biografi Penulis



Amru Rosyada. Lahir pada tanggal 22 Mei 1986, menamatkan pendidikan dasar sampai pendidikan menengah akhir di kota Ngawi kemudian terdampar di Jogja mengambil program Diploma tiga Teknik Elektro Universitas Gadjah Mada dan Sekarang masih menamatkan Strata satu di IlmuKomputer Universitas Gadjah Mada.