Annotation

Ngọc Lục

Annotation là gì?

Annotation là gì

Annotation trong Java là một thẻ đại diện cho metadata, nó thường đi kèm với lớp, interface, phương thức hoặc các biến để chỉ ra một số thông tin bổ sung có thể được sử dụng bởi trình biên dịch Java.

Annotation sử dụng để cung cấp thông tin bổ sung, do đó nó là một tùy chọn thay thế cho XML và marker interface

Đặc điểm của Annotation

- Annotation bắt đầu với @
- Annotation không làm thay đổi hoạt động của một chương trình đã compile
- Annotation giúp liên kết thông tin với các phần tử của chương trình
- Annotation không phải là chú thích thuần túy vì chúng có thể thay đổi cách trình biên dịch xử lý chương trình

Mục đích sử dụng Annotation

- Chỉ dẫn cho trình biên dịch
- Chỉ dẫn trong thời điểm biên dịch
- chỉ dẫn trong thời gian chạy

Các loại annotation

Marker Annotation

Nhằm mục đích để đánh dấu một khai báo. Các annotation không chứa bất kỳ thành viên nào và không chứa bất kỳ dữ liệu nào. Có thể kể đến như **@Override**

Ví dụ: @TestAnnotation

Single value annotation

Annotation này chỉ chứa một thành viên và cho phép dạng viết tắt dạng xác định giá trị thành viên. Khi chúng ta áp dụng chú thích này, chúng ta chỉ cần chỉ định giá trị cho thành viên mà không cần chỉ định tên thành viên. Tuy nhiên, để sử dụng các viết tắt này phải có một giá trị cho tên của thành viên

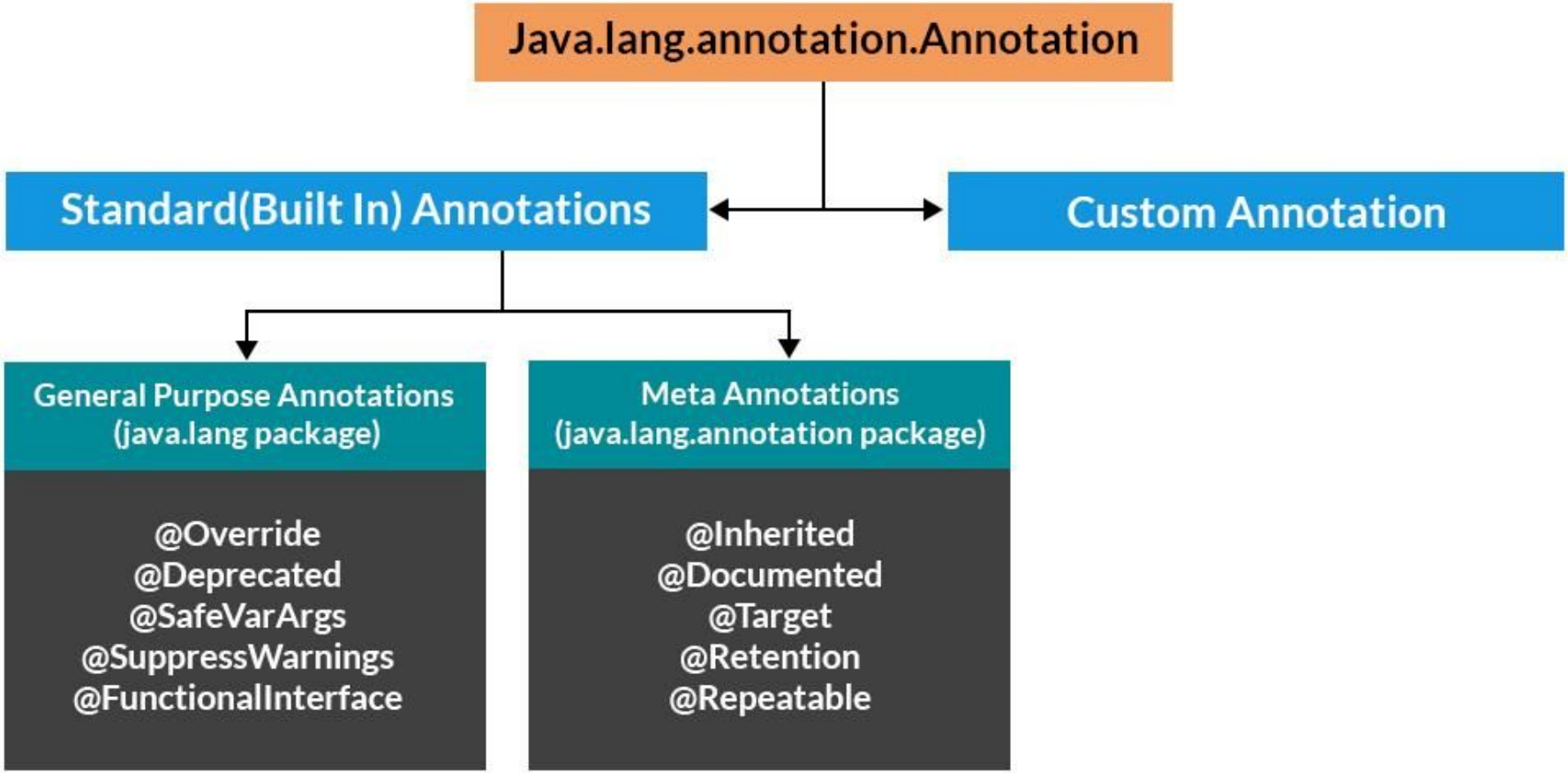
Ví dụ: @TestAnnotation(“value")

Multi-value annotation

Các annotation này bao gồm nhiều thành viên dữ liệu, tên, giá trị, pair

Ví dụ: @TestAnnotation(value1 = 10, value2 = “Eri”)

Built-in Java Annotation


Java.lang.annotation.Annotation
Standard(Built In) Annotations
Custom Annotation
General Purpose Annotations
(java.lang package)
@Override
@Deprecated
@SafeVarArgs
@SuppressWarnings
@FunctionalInterface
Meta Annotations
(java.lang.annotation package)
@Inherited
@Documented
@Target
@Retention
@Repeatable

@Deprecated

@Deprecated là một marker annotation, nó chỉ ra rằng một class hoặc method bị lỗi thời và không nên sử dụng nữa.

Bạn có thể tìm thấy nó trong một số class và phương thức có sẵn trong Java, ví dụ như: class java.util.Date, có thể tìm thấy một số constructor, method bị đánh dấu @Deprecated

```
@Deprecated
public Date(int year, @MagicConstant(...) int month, int date) {
    this(year, month, date, hrs: 0, min: 0, sec: 0);
}
```

```
@Deprecated
public static long parse(String s) {
    int year = Integer.MIN_VALUE;
    int mon = -1;
    int mday = -1;
    int hour = -1;
```

@Override

Annotation **@Override** là một marker annotation, nó được sử dụng cho các phương thức ghi đè của phương thức trong class cha

Annotation này là không bắt buộc nhưng khi thực hiện ghi đè phương thức ta nên sử dụng annotation này để đánh dấu phương thức, việc này giúp code dễ đọc và dễ bảo trì

@SuppressWarnings

Annotation này được sử dụng để hướng dẫn trình biên dịch bỏ qua những cảnh báo cụ thể

Java chia cảnh báo ra thành hai dạng: deprecated và unchecked

Constructor của lớp java.util.Date bị cảnh báo là không nên sử dụng

```
public class Main {
    public static void main(String[] args) {
        Date date = new Date( year: 2022, Calendar.JUNE, date: 12);
        System.out.println(date);
    }
}
```

Áp dụng annotation @SuppressWarnings

```
    @SuppressWarnings("deprecation")
    public static void main(String[] args) {
        Date date = new Date( year: 2022, Calendar.JUNE, date: 12);
        System.out.println(date);
    }
```

@SuppressWarnings

Các tham số được truyền trong @SuppressWarnings

- deprecation
- unchecked
- rawtypes

@Documented

Annotation này chỉ ra rằng annotation mới nên được bao gồm trong tài liệu Java được tạo ra bởi các công cụ tạo tài liệu Java

@Target

Sử dụng để chú thích cho annotation khác và annotation đó sẽ được sử dụng trong phạm vi nào

Element Types	Where the annotation can be applied
TYPE	class, interface or enumeration
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variables
ANNOTATION_TYPE	annotation type
PARAMETER	parameter

@Retention

Annotation @Retention dùng để chú thích mức độ tồn tại của một annotation nào đó

Có 3 mức nhận thức tồn tại được chú thích:

- RetentionPolicy.SOURCE
- RetentionPolicy.CLASS
- RetentionPolicy.RUNTIME

@Inherited

Annotation chỉ ra rằng loại chú thích có thể được thừa kế từ lớp cha (mặc định là false). Khi người dùng truy vấn kiểu annotation của lớp con và lớp con không có annotation cho kiểu này thì lớp cha của lớp được truy vấn cho loại annotation này sẽ được gọi.
Annotation này chỉ áp dụng cho khi báo các lớp

Custom Annotation

Custom Annotation

Ta sử dụng @interface để khai báo custom annotation

Marker Annotation

Annotation nào không có phương thức thì được gọi là Marker Annotation. Ví dụ:

```
@Target({ElementType.METHOD,ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface MarkerAnnotation {
}
```

Single Value Annotation

Annotation có một phương thức, được gọi là Single Value Annotation. Ví dụ:

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface SingleValueAnnotation {
    int value() default 0;
}
```

Multi Value Annotation

Annotation có từ 2 phương thức trở lên được gọi là Multi Value Annotation. Ví dụ:

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface MultiValueAnnotation {
    int value1() default 0;
    String value2() default "abc";

    String value3() default "xyz";
}
```