# Cơ bản về Annotation

## Định nghĩa

Annotation tạm dịch là “Chú thích”, theo nghĩa là ghi chú trong code, nhưng bản chất nó là một dạng siêu dữ liệu (metadata) cung cấp chương trình khác trong chương trình đang viết.

Siêu dữ liệu có thể hiểu đơn giản là dữ liệu dùng để để mô tả dữ liệu.

## Ví dụ về Annotation

Ví dụ:

* @Entity đây là annotation đơn giản nhất cho một lớp
* @Override

void mySuperMethod() { ... }

Annotation được hiểu cho việc kế thừa method (phương thức) từ lớp cha

## Cách sử dụng Annotation

Annotation sử dụng cho các việc sau:

* Cung cấp thông tin cho trình biên dịch để nhằm để phát hiện lỗi, cảnh báo.
* Xử lý tác vụ trong quá trình biên dịch và khởi chạy : Các phần mềm có thể đọc Annotation để tạo mã và các tệp XML,..
* Sử dụng trong lúc chạy (Runtime): Một số Annotation có sẵn để sử dụng kiểm tra Runtime

Annotation được gọi từ package:  “java.lang” hoặc “java.lang.annotation”.

Về cơ bản Annotation thường hay là một dòng chú thích độc lập, từ phiên bản Java 8 trở đi, thì Annotation có thể đứng cùng dòng với code. Ví dụ:

* new @Interned MyObject(); khởi tạo
* myString = (@NonNull String) str;

Annotation có thể sử dụng trên nhiều dòng

Ngoài ta chúng ta có thể tạo Annotation của riêng ứng dụng chúng ta. Hãy xem cách tạo và sử dụng

# Tạo Annotation riêng

Rất nhiều Annotation thay thế cho comments trong code.

Gỉa sử một phần mềm yêu cầu phải có Thông tin người tạo class đó trong code thì ta nên xây dựng annotation cho toàn ứng dụng. Hãy xem ví dụ sau:

|  |
| --- |
| **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  } |

Ta hoàn toàn có thể xây dựng Annotation để thay thế các comment trên.

Ta xác định cú pháp như sau: Author, Date, Current revision, Last modified, By, Reviewers

Sau khi xác định cú pháp ta xây dựng một Interface

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

Tất cả các chú thích là dạng Interface. Trong tài liệu này các bạn có thể chưa cần hiểu interface là gì

Định nghĩa **@interface** được hiểu là ta đang khai báo Annotation, **@** có thể hiểu viết tắt cho Annotation.

Như vậy đã tạo được Annotation vậy việc sử dụng ra sao. Rất đơn giản, chúng ta sử dụng @ClassPreamble và điền các thông số.

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

# Các loại Annotation

Sau đây là các annotation đã được định nghĩa trước và ta chỉ cần sử dụng

Một tập annotation được định nghĩa sẵn trong Java SE API. Một số annotation sử dụng cho trình biên dịch và một số loại chú thích được áp dụng cho các chú thích khác.

## Các loại annotation được sử dụng bởi Java

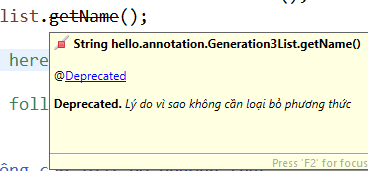
Các loại chú thích được xác định trước được định nghĩa trong java.lang là @Deprecated, @Override và @SuppressWarnings.

### @Deprecated

### Annoation @Deprecated đánh đấu rằng phương thức, class,.. này không được công nhận và không nên sử dụng nữa. Nó thông báo cho người dùng rằng nó có thể được gỡ bỏ trong các phiên bản trong tương lai. Vì vậy, tốt hơn là không sử dụng các phương thức như vậy.

|  |
| --- |
| // Javadoc comment follows  /\*\*  \* **@deprecated**  \* Lý do vì sao không cần loại bỏ phương thức  \*/  @Deprecated  **private** String ~~getName~~() {  **return** "";  } |

Trong ví dụ bạn thấy ~~getName~~ là phương thức khuyến nghị không nên sử dụng. Và nó được comment lý do loại bỏ. Khi sử dụng phương thức này ta sẽ thấy như sau



Các lý do sẽ hiển thị để bạn có thể thấy khi đưa chuột vào phương thức trong Eclipse.

### @Override

Với @Override nhằm khai báo cho trình biên dịch biết phần tử này ghi đè lên một phần tử cùng tên trong Class khác, chúng ta thường gọi là lớp cha(SupperClass). Từ khóa thể hiện tính kế thừa của Java

Tuy annotation này không thực sự cần thiết khi ghi đè, nhưng nó giúp ngăn ngừa lỗi. Nếu một phương thức được đánh dấu bằng @Override không ghi đè chính xác phương thức trong một trong các siêu lớp của nó, trình biên dịch sẽ tạo ra một lỗi.

|  |
| --- |
| // mark method as a superclass method  // that has been overridden  @Override  **public** String getDate() {  **super**.getDate();  **return** "";  } |

### @SuppressWarnings

## Annotation sử dụng cho Annotation khác

### @Target

### @Retention

### @Inherited

### @Documented