Java Type and Repeating Annotations

Java Type Annotations

Java 8 has included two new features repeating and type annotations in its prior annotations topic. In early Java versions, you can apply annotations only to declarations. After releasing of Java SE 8 , annotations can be applied to any type use. It means that annotations can be used anywhere you use a type. For example, if you want to avoid NullPointerException in your code, you can declare a string variable like this:

1. @NonNull String str;

Following are the examples of type annotations:

1. @NonNull List<String>
2. List<@NonNull String> str
3. Arrays<@NonNegative Integer> sort
4. @Encrypted File file
5. @Open Connection connection
6. **void** divideInteger(**int** a, **int** b) **throws** @ZeroDivisor ArithmeticException

**Note - Java created type annotations to support improved analysis of Java programs. It supports way of ensuring stronger type checking.**

Java Repeating Annotations

In Java 8 release, Java allows you to repeating annotations in your source code. It is helpful when you want to reuse annotation for the same class. You can repeat an annotation anywhere that you would use a standard annotation.

For compatibility reasons, repeating annotations are stored in a container annotation that is automatically generated by the Java compiler. In order for the compiler to do this, two declarations are required in your code.

1. Declare a repeatable annotation type
2. Declare the containing annotation type

1) Declare a repeatable annotation type

Declaring of repeatable annotation type must be marked with the @Repeatable meta-annotation. In the following example, we have defined a custom @Game repeatable annotation type.

1. @Repeatable(Games.**class**)
2. @interfaceGame{
3. String name();
4. String day();
5. }

The value of the @Repeatable meta-annotation, in parentheses, is the type of the container annotation that the Java compiler generates to store repeating annotations. In the following example, the containing annotation type is Games. So, repeating @Game annotations is stored in an @Games annotation.

2) Declare the containing annotation type

Containing annotation type must have a value element with an array type. The component type of the array type must be the repeatable annotation type. In the following example, we are declaring Games containing annotation type:

1. @interfaceGames{
2. Game[] value();
3. }

**Note - Compiler will throw a compile-time error, if you apply the same annotation to a declaration without first declaring it as repeatable.**

Java Repeating Annotations Example

1. // Importing required packages for repeating annotation
2. **import** java.lang.annotation.Repeatable;
3. **import** java.lang.annotation.Retention;
4. **import** java.lang.annotation.RetentionPolicy;
5. // Declaring repeatable annotation type
6. @Repeatable(Games.**class**)
7. @interfaceGame{
8. String name();
9. String day();
10. }
11. // Declaring container for repeatable annotation type
12. @Retention(RetentionPolicy.RUNTIME)
13. @interfaceGames{
14. Game[] value();
15. }
16. // Repeating annotation
17. @Game(name = "Cricket",  day = "Sunday")
18. @Game(name = "Hockey",   day = "Friday")
19. @Game(name = "Football", day = "Saturday")
20. **public** **class** RepeatingAnnotationsExample {
21. **public** **static** **void** main(String[] args) {
22. // Getting annotation by type into an array
23. Game[] game = RepeatingAnnotationsExample.**class**.getAnnotationsByType(Game.**class**);
24. **for** (Gamegame2 : game) {    // Iterating values
25. System.out.println(game2.name()+" on "+game2.day());
26. }
27. }
28. }

OUTPUT:

Cricket on Sunday

Hockey on Friday

Football on Saturday

# References

https://www.javatpoint.com/java-8-type-annotations-and-repeating-annotations