Tech Tutorials

Tutorials and posts about Java, Spring, Hadoop and many more. Java code examples and interview questions. Spring code examples.

Wednesday, 29 November 2017

Functional Interface Annotation in Java

In the post Functional Interfaces in Java we have already seen that functional interfaces are those interfaces that have only one abstract method.

Java 8 also introduced an annotation **@FunctionalInterface** to be used with functional interfaces. Annotating an interface with @FunctionalInterface in Java indicates that an interface type declaration is intended to be a functional interface.

It is **not mandatory to mark functional interface** with @FunctionalInterface annotation, it is more of a best practice to do that and also gives a surety that no other abstract method will be added **accidentally to the functional interface**. Because it will result in compiler error if any other abstract method is added to a functional interface which is annotated with @FunctionalInterface annotation.

Let's see it with some examples what is permitted and what is not with @FunctionalInterface annotation in Java.
A **valid example** of a functional interface -

```
@FunctionalInterface
public interface IMyFuncInterface {
  public void getValue();
}
```

Note that in Java 8 default methods and static methods are also added in interface which means interface can have a method with default implementation and static methods in Java 8. In a functional interface there may be one or more default methods/static methods but there should be only one abstract method. It is ok to have a functional interface like following.

```
@FunctionalInterface
public interface IMyFuncInterface {
    int func(int num1, int num2);
    // default method
    default int getValue(){
        return 0;
    }
}
```

When you annotate an interface with @FunctionalInterface, if more than one abstract method is defined it will throw a **compiler error**.

```
@FunctionalInterface
public interface IMyFuncInterface {
  public void getValue();
  // Second abstract method so compiler error
  public void setValue();
}
```

A functional interface can specify **Object class public methods** too in addition to the abstract method. That interface will still be a **valid functional interface**. The public Object methods are considered implicit members of a functional interface as they are automatically implemented by an instance of functional interface.

As example- This is a valid functional interface

```
@FunctionalInterface
interface IFuncInt {
    int func(int num1, int num2);
```

```
    // default method
    default int getValue(){
        return 0;
    }
    public String toString();
    public boolean equals(Object o);
}
```

That's all for this topic **Functional Interface Annotation in Java**. If you have any doubt or any suggestions to make please drop a comment. Thanks!

Related Topics

1. Lambda Expressions in Java 8
2. How to Fix The Target Type of This Expression Must be a Functional Interface Error
3. Method Reference in Java 8
4. Lambda Expression as Method Parameter
5. Java Lambda Expressions Interview Questions

You may also like -

- fail-fast Vs fail-safe iterator in Java
- How to iterate a Hash map of arraylists of String in Java?
- strictfp in Java
- static import in Java
- Synchronization in Java multithreading
- Deadlock in Java multi-threading
- Difference between Checked exception & Unchecked exception
- Externalizable Interface in Java

>>>Go to Java Advance Topics Page | Go to Lambda Expression Tutorial Page>>>

Posted by Anshudeep Bajpai at 21:42

Labels: Java 8, lambda expression

2 comments:

Unknown 27 October 2017 at 01:01

Hi Anshudeep, the only difference between abstract and public methods in FunctionalInterface is access specifier?

Reply

Replies

Functional interfaces in Java 8 29 October 2017 at 22:32

Looks like you have some confusion here in an interface all methods are by default public. The interface body can contain abstract methods, default methods, and static methods. All abstract, default, and static methods in an interface are implicitly public, in fact you can omit the public modifier altogether if you want.

Reply

Enter your comment...

Comment as: lemon_leave (Go

Sign out

Publish Preview

Notify me

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Disclaimer Privacy Policy

Ethereal theme. Powered by Blogger.