Java 8 之 函数式接口

作者：wei
原文链接：https://ld246.com/article/1505579670669
来源网站：链滴

Java 8 引入了函数式接口（Functional Interfaces）的概念：如果一个接口只定义了唯一一个抽象方，那么这个接口就成为函数式接口。同时，Java 8中还引入了一个新的注解：@FunctionalInterface可以把这个注解放在一个接口前，用于表示这个接口是一个函数式接口。这个注解是非必须的，只要口只定义了唯一一个抽象方法，虚拟机就会自动判断出该接口为一个函数式接口，不过最好在定义的数式接口上加上 @FunctionalInterface 注解。对于加了 @FunctionalInterface 注解的接口，Java 译器会在编译时对接口进行检查，如果发现该接口中定义的抽象方法不是一个，那么编译器会报错，样就可以保证该接口的定义符合预期。

java.lang.Runnable就是一个典型的函数式接口：

```
@FunctionalInterface
public interface Runnable {
    void run();
}
```

另外，为了避免开发者创建大量的仅为 Lambda 表达式服务的函数式接口，Java 8 在 java.util.functin 中增加了不少新的函数式通用接口。例如：

Function<T,R> :将 T 作为输入，返回 R 作为输出，他还包含了
其他函数组合的默认方法。

Predicate<T> ： 将 T 作为输入，返回一个布尔值作为输出，该
口包含多种默认方法来将 Predicate 组合成其他复杂的逻辑（与、或、非）。

Consumer<T> ： 将 T 作为输入，不返回任何内容，表示在单
参数上的操作。

简单讲解一下Function<T,R>
口，它的定义如下：

```
@FunctionalInterface
public interface Function {
    R apply(T var1);

    default  Function compose(Functionsuper V, ? extends T> var1) {
        Objects.requireNonNull(var1);
        return (var2) -> {
            return this.apply(var1.apply(var2));
        };
    }

    default  Function andThen(Functionsuper R, ? extends V> var1) {
        Objects.requireNonNull(var1);
        return (var2) -> {
            return var1.apply(this.apply(var2));
        };
    }

    static  Function identity() {
        return (var0) -> {
            return var0;
        };
    }
}
```

可以看到Function接口中只有一个抽象方法R apply(T var1) 这个方法唯一一个待实现的方法（抽象法），这个方法将Function对象应用到输入的参数上，然后返回计算结果。

default Function compose(Functionsuper V, ? extends T> var1) 为一个默认方法，如不需要重写以保留其默认实现。它返回一个先执行传入的函数对象的apply方法再执行当前函数对象的apply方法函数对象。

default Function andThen(Functionsuper R, ? extends V> var1) 方法与它的上一个方法很相似它返回一个先执行当前函数对象的apply方法再执行传入的函数对象的apply方法的函数对象。

static Function identity() 为一个静态方法，它返回一个执行了apply()方法之后只会返回输入参数即apply()方法中直接输出输入的数据）的函数对象。