Java 8 之 Lambda 表达式

作者：wei
原文链接：https://ld246.com/article/1505576789714
来源网站：链滴

Lambda表达式（闭包）的支持可以说是Java 8带来的最大的、最激动人心的改变。Lambda允许把数作为一个方法的参数（函数作为参数传递进方法中），或者把代码看成数据。很多JVM平台上的语自诞生之初就支持lambda表达式，如groovy、scala等，但是Java开发者却一直都只能用匿名类来代，现在，Java 8终于把Lambda表达式引入了，Java开发者现在多了一个选择。

但是在Java中，函数是不能直接作为参数传递的，但是Lambda表达式却是一个匿名函数，那么Java是如何解决这个问题的呢？为了让现有的功能与Lambda表达式良好兼容，Java 8中引入了函数式接口概念。函数式接口指的是只有一个抽象方法的接口，这样的接口可以隐式转换为Lambda表达式。

Lambda表达式由三个部分组成：第一部分为一个括号内用逗号分隔的形式参数，参数是函数式接口面方法的参数；第二部分为一个箭头符号：->；第三部分为方法体，可以是表达式和代码块。语法如：

1.方法体为表达式，该表达式的值作为返回值返回。

```
(parameters) -> expression
```

2. 方法体为代码块，必须用 {} 来包裹起来，且需要一个 return 返回值，但若函数式接口里面方法回值是 void，则无需返回值。

```
(parameters) -> { statements; }
```

如果参数只有一个，那么参数列表的括号可以省略，这样Lambda表达式就简化为：

```
parameter -> expression
```

注意，Lambda表达式中参数的类型可以由编译器推断出来，无需显式指定（当然，显式指定也是可的），下面是一个在foreach中使用Lambda表达式的列子，可以看到，参数无需指定类型：

```
Arrays.asList( "a", "b", "c", "d" ).forEach( w -> System.out.println( w ) );
```

其实对比起Java中传统的匿名类，Lambda表达式的一个重要特点的代码简洁，下面的代码是一个使Lambda表达式来创建并启动一个Java线程的例子：

```
new Thread(() -> System.out.println("Hello, Lambda Expressions！ ")).start();
```

对，你没有看错，使用Lambda来创建一个线程就是如此简单，对比以往传统创建线程的方法，显然简洁得多，以下是使用传统的匿名类来创建并启动一个线程的方法：

```
new Thread(new Runnable() {
    @Override
  public void run() {
        System.out.println("I create a thread !");
    }
}).start();
```

是不是对如此简洁的写法心动了？其实Lambda表达式带来的不仅仅是更加简洁的代码，它还带来了高的效率，并且还在Java开发中带来了一个全新的概念——函数式编程。很快，你将会知道Java 8中多的新特性都是围绕着Lambda表达式而来的，如需了解更多关于Lambda表达式的细节，可以参考方文档。