Iteye关于Java8的简明教程

<http://www.iteye.com/magazines/129-Java-8-Tutorial>

#### 目 录 [[ - ]](http://www.iteye.com/magazines/129-Java-8-Tutorial)

1. [允许在接口中有默认方法实现](http://www.iteye.com/magazines/129-Java-8-Tutorial#548)
2. [Lambda表达式](http://www.iteye.com/magazines/129-Java-8-Tutorial#549)
3. [函数式接口](http://www.iteye.com/magazines/129-Java-8-Tutorial#550)
4. [方法和构造函数引用](http://www.iteye.com/magazines/129-Java-8-Tutorial#551)
5. [Lambda的范围](http://www.iteye.com/magazines/129-Java-8-Tutorial#552)
6. [内置函数式接口](http://www.iteye.com/magazines/129-Java-8-Tutorial#553)
7. [Streams](http://www.iteye.com/magazines/129-Java-8-Tutorial#554)
8. [Parallel Streams](http://www.iteye.com/magazines/129-Java-8-Tutorial#555)
9. [Map](http://www.iteye.com/magazines/129-Java-8-Tutorial#556)
10. [时间日期API](http://www.iteye.com/magazines/129-Java-8-Tutorial#557)
11. [Annotations](http://www.iteye.com/magazines/129-Java-8-Tutorial#558)
12. [总结](http://www.iteye.com/magazines/129-Java-8-Tutorial#559)
13. lambda
    1. 定义：函数式接口
    2. 作用范围
       1. 可以访问外部的final常量
       2. 可以访问外部仅定义一次之后不再改变的变量（因为不再改变，所以lambda表达式把它当常量处理）
       3. 类外部定义的Lambda表达式内部不能外部变量值
       4. 如果lambda表达式在类内部定义，可以有函数内部变量的读写权限。
       5. 接口的默认方法不能被lambda表达式访问
14. 内置函数式接口
    1. 函数式接口定义：是个接口，只包含一个抽象方法，那么它就是函数式接口，我们可以在任意函数式接口上使用 @FunctionalInterface 检查它是否是一个函数式接口。函数式接口里是可以包含默认方法、静态方法，他们不是抽象方法；也可以包含Java.lang.Object里的public方法，因为任何一个类都继承Object类，包含了来自java.lang.Object里对这些抽象方法的实现，也不属于抽象方法；函数式接口里允许子接口继承多个父接口，但每个父接口中都只能存在一个抽象方法，且必须的相同的抽象方法。
    2. 四大核心函数式接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数式接口 | 方法 | 参数类型 | 返回类型 | 作用 |
| Consumer<T> 消费  型接口 | void accept(T t) | T | void | 对T类型的参数进行操作 |
| Supplier<T> 供给型接口 | T get() | 无 | T | 操作数据,返回T类型的结果 |
| Function<T, R> 函数型接口 | R apply(T t) | T | R | 对T类型参数进行操作，并返回R类型的结果 |
| Predicate<T> 断定型接口 | boolean test(T t) | T | boolean | 确定T类型参数是否满足某约束，并返回boolean值 |
| //Consumer<T> 消费型接口  @Test  public void test1(){  Consumer<String> c = (x) -> System.out.println("hello:"+x+"!");  c.accept("Java");  } | | | | | |
| // Supplier<T> 供给型接口  @Test  public void test2(){  Supplier<String> s = () -> "hello,beautiful girl";  String str = s.get();  System.out.println(str);  } | | | | | |
| //Function<T,R> 函数性接口  @Test  public void test3(){  Function<String, Integer> f= (x) -> x.length();  Integer len = f.apply("hello");  System.out.println(len);  } | | | | | |
| //Predicate<T> 断言型接口  @Test  public void test4(){  Predicate<String> p = (x) -> x.length()>5;  boolean b = p.test("hello Java");  System.out.println(b);  } | | | | | |

1. 流对象
   1. Stream（）返回对象的流视图，对返回视图的所有操作都不会影响原对象
   2. ParallelStream返回并行流对象
   3. Sort（）对流对象进行排序，但是并不影响原对象
      1. 接受一个compartor型参数
   4. Filter（）对流对象进行过滤，接受一个predicate参数,返回通过过滤器的对象
   5. Foreach（）逐个对流对象进行操作，接受一个function型参数
   6. match匹配操作有多种不同的类型，都是用来判断某一种规则是否与流对象相互吻合的。所有的匹配操作都是终结操作，只返回一个boolean类型的结果。
   7. map（）相当于foreach（）对每个流对象进行一些处理，是一个终结操作
   8. count（）计算现有流对象的个数，是终结操作。
2. 注解
   1. 一个类上可以有多个注解了，不需要用注解容器