链滴

Lambda 表达式之 foreach

作者：zml2015

原文链接：https://ld246.com/article/1495521493698

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

常规情况下我们一般是用简单for循环或者增强for循环

```
        List<String> list = new ArrayList<>();
         list.add("123");
        //第一种
         for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i));
        }
         //第二种
         for (String string : list) {
            System.out.println(string);
        }
```

使用jdk1.8之后，简单类型封装类可以这样写输出：

```
//no.1
list.forEach((String str) -> {
    System.out.println(str);
});

//no.2
list.forEach(str -> {
    System.out.println(str);
});

//no.3
list.forEach(str -> System.out.println(str));

//no.4
list.forEach(System.out::println);

//no.5
list.forEach(new MyConsumer());
```

MyConsumer类如下,主要是接口的实现

```
class MyConsumer implements Consumer<String>{

    @Override
    public void accept(String str) {
        System.out.println(str);
    }
}
```

何为简单类型封装类?

- Integer
- Short
- Long
- Byte
- Double

- Float
- String
- Boolean
- ...

如果是复杂类型（自定义类型），则上述方式则不全都能适用了，但可以这样

```
MongoDatabase database = MongoDBJDBC.getMongoDB();
MongoCollection<Document> collection = database.getCollection("user");
BasicDBObject query = new BasicDBObject();
FindIterable<Document> users = collection.find(query).sort(new BasicDBObject("age", 1));
```

1. 第一种,可以进行多步处理操作，第一种与第二种方式的区别，类似于if后面加大括号与不加的区别

```
users.forEach((Document block) -> {String str = block.getString("key");
System.out.println(str);
System.out.println(block.getObjectId("_id"));});
```

2. 第二种，仅简单的处理处理一次

java

```
users.forEach((Document block) -> System.out.println(block.getObjectId("_id")));
```

* 第三种，实现Consumer接口

```java
class DocumentConsumer implements Consumer<Document>{

    @Override
    public void accept(Document t) {
    System.out.println(t);

    }

}

users.forEach(new DocumentConsumer());
```