# Java lambda表达式

- What?Why?How?

# 本章介绍

- Lambda表达式的介绍
- Lambda表达式的使用
- 函数式接口
- Lambda表达式的原理

# Lambda表达式的介绍

- Lambda表达式是 Java8 中最重要的新功能之一。使用 Lambda 表达式可以替代只有一个抽象函数的接口实现，告别匿名内部类，代码看起来更简洁易懂。Lambda表达式同时还提升了对集合、框架的迭代、遍历、过滤数据的操作。

# Lambda表达式的特点

1：函数式编程

2：参数类型自动推断

3：代码量少，简洁

# Lambda表达式案例

```
public class Test {
    public static void main(String[] args) {
        //匿名内部类方式
        new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("run.........");
            }
        }).start();
        //lambda表达式方式
        new Thread(()->{System.out.println("run........");}).start();
    }
}
```

# Lambda表达式案例

```
public class Test {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("java","python","scala","javascript");
        //普通匿名内部类方式
        Collections.sort(list, new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                return o1.length()-o2.length();
            }
        });
        //lambda方式
        Collections.sort(list,(a,b)->a.length()-b.length());
        list.forEach(System.out::println);
    }
}
```

# Lambda表达式应用场景

任何有**函数式接口**的地方

# 函数式接口

只有一个抽象方法（Object类中的方法除外）的接口是函数式接口

# 函数式接口

Supplier 代表一个输出

Consumer 代表一个输入
BiConsumer 代表两个输入

Function 代表一个输入，一个输出（一般输入和输出是不同类型的）
UnaryOperator 代表一个输入，一个输出（输入和输出是相同类型的）

BiFunction 代表两个输入，一个输出（一般输入和输出是不同类型的）
BinaryOperator 代表两个输入，一个输出（输入和输出是相同类型的）