BLS能力中心技术分享

朱文魁

UNITED 联影 IMAGING

背景

1.java版本（1.0->5.0->8.0->14），jdk1.8 lambda表达式

2. lambda表达式 是什么

Lambda java8的一个新语法，允许把函数作为一个方法的参数（函数作为参数传递进方法中），可以理解为简洁的匿名函数

3. lambda能带来什么

1）java 的一个大的改进版本（python，js），代码变的更加简洁紧凑，提高开发效率
2）延迟执行（先定义，在适当的时候再执行）
3）通过lambda 我们又可以封装一堆好用的tools，比如官方的 Stream Api
4）让你的代码看起来高大上

4. 现状：

项目中虽然已经是8版本了，但是lambda表达式在 项目中很少有用到。

分享内容

1 lambda表达式语法
2 函数式接口
3 实例说明
4 StreamApi
5 lambda & Stream实现原理
6 ControlHelper tools
7 CatchUtils tools

lambda表达式 ->

```
// 1. 不需要参数,返回值为 5
() -> 5

// 2. 接收一个参数(数字类型),返回其2倍的值
x -> 2 * x

// 3. 接收2个参数(数字),并返回他们的差值
(x, y) -> x – y

// 4. 接收2个int型整数,返回他们的和
(int x, int y) -> x + y

// 5. 接收一个 string 对象,并在控制台打印,不返回任何值(看起来像是返回void)
(String s) -> System.out.print(s)
```

Lambda 方法引用 ::

方法引用可以理解为lambda表达式的快捷写法

类型	语法	对应的Lambda表达式
静态方法引用	类名::staticMethod	(args) -> 类名.staticMethod(args)
实例方法引用	inst::instMethod	(args) -> inst.instMethod(args)
对象方法引用	类名::instMethod	(inst,args) -> 类名.instMethod(args)
构建方法引用	类名::new	(args) -> new 类名(args)

函数式接口

就是一个接口，只有一个抽象方法，可隐式转换成lambda表达式

函数式接口	参数类型	返回类型	用途
Consumer<T>(消费型接口)	T	void	对类型为T的对象应用操作。void accept(T t)
Supplier<T>(供给型接口)	无	T	返回类型为T的对象。 T get();
Function<T, R>(函数型接口)	T	R	对类型为T的对象应用操作并返回R类型的对象。R apply(T t);
Predicate<T>(断言型接口)	T	boolean	确定类型为T的对象是否满足约束。boolean test(T t);

StreamApi

1 概念

1. Stream 是java8中新增加的一个接口，将要处理的元素集合看作一种流， 流在管道中传输， 并且可以在管道的节点上进行处理， 比如筛选， 排序，聚合等。
2. 分为中间操作和终结操作
3. Lambda最佳实践

2 api简介

Stream操作分类		
中间操作(Intermediate operations)	无状态(Stateless)	unordered() filter() map() mapToInt() mapToLong() mapToDouble() flatMap() flatMapToInt() flatMapToLong() flatMapToDouble() peek()
	有状态(Stateful)	distinct() sorted() sorted() limit() skip()
结束操作(Terminal operations)	非短路操作	forEach() forEachOrdered() toArray() reduce() collect() max() min() count()
	短路操作(short-circuiting)	anyMatch() allMatch() noneMatch() findFirst() findAny()

StreamApi 实例

3 u课堂实例代码演示

feature-zwk

FileBackController

AliyunTreeService

grouptest

lambda & Stream实现原理

1. Lambda 实现

1）编译阶段会生成对应的lambda函数 javap -p -v
2）使用invokedynamic 指令 MethodHandle （相对于 反射的 Method 更轻量级）动态命令来生成内部类
3）内部类直接调用 父类中的lambda 函数

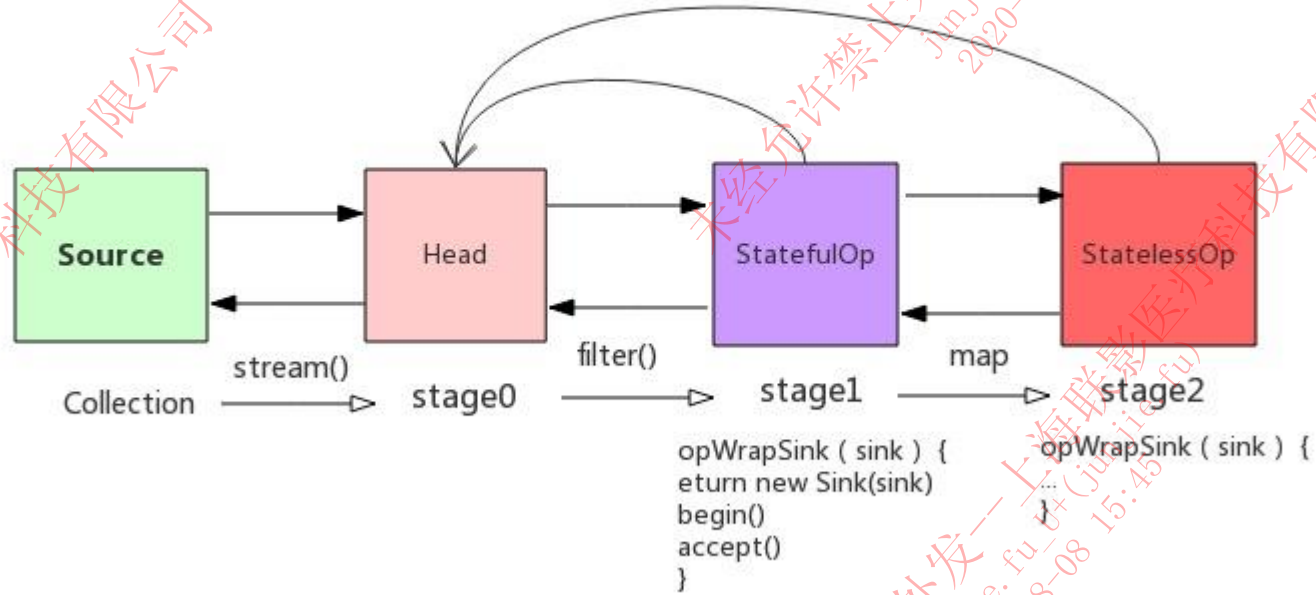
2 Stream 实现

Stream.of(1,2,3).filter(c->c!=1).map(String::valueOf).collect(Collectors.toList());
1.多次迭代 时间复杂度增加，内存消耗大

lambda & Stream实现原理

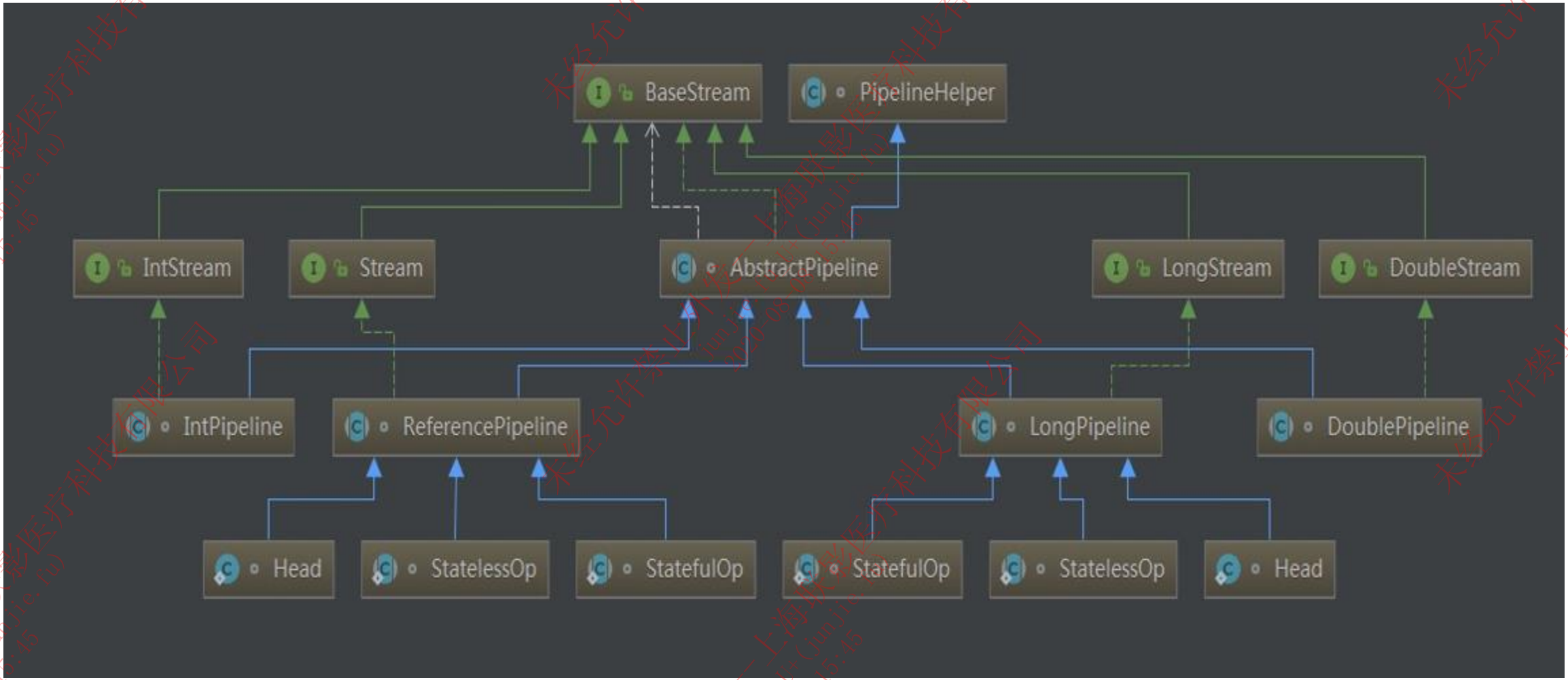
2.Stream 中case调用链
ReferencePipeline（of）--> new StatelessOp(previousStage=of) --> new StatelessOp(previousStage=filter) -->
collect(ReduceOp 终结操作) -->通过递归调用opWrapSink 生成sink
-->反向传递当前sink -->setDownstream -->最终得到第一个sink filter操作
--> copyInto开始真正执行sink --> 后续操作

```
Stream.of(1,2,3).filter(c->c!=1).map(String::valueOf).collect(Collectors.toList());
```



lambda & Stream实现原理

1. Stream 接口继承关系图



CatchUtil tools

1. 代码演示

2. 能带来什么

1. 可以和业务具体返回一起使用 简化 异常处理流程
2. 可以处理 check 异常，改完 手动 threw
3. 自动记录日志

3. U课堂使用实例

Controller tools

1. 代码演示

2. 能带来什么

1. 可以自动使用Validator 进行参数验证
2. 可以扩展返回内容，如接口执行时间/监控/日志
3. 自动根据异常类型配置返回code等信息

3. U课堂使用实例

结论

Q&A