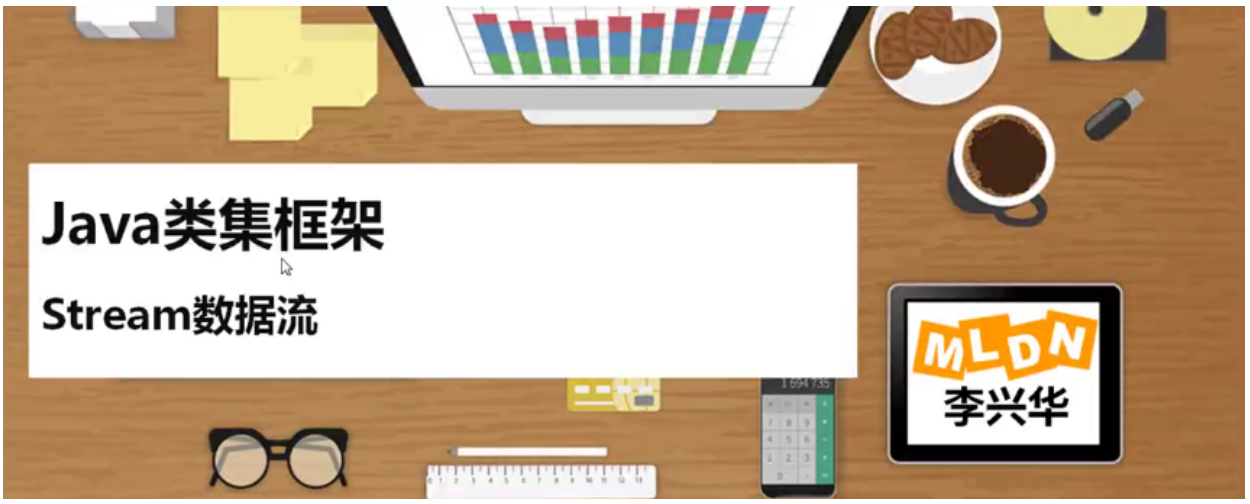

博客： https://www.cnblogs.com/HOsystem/p/14116443.html

2、具体内容

从JDK1.8开始，由于已经进入到了大数据的时代,所以在类集里面也支持有数据的流式分析处理操作,为此就专门提供了一个Stream的接口,同时在collection接口里面也提供有为此接口实例化的方法;

·获取Stream接口对象：public default Stream<E> stream();

■Stream基础操作

Stream主要功能是进行数据的分析处理，同时主要是针对于集合中的数据进行分析操作。

范例：Stream的基本操作

```
package cn.mldn.demo;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.stream.Stream;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		List<String> all = new ArrayList<String>();
		Collections.addAll(all, "Java", "JavaScript", "Python", "Ruby", "Go");
		Stream<String> stream = all.stream() ; // 可以获得Stream接口对象
		// 要求将每一个元素的字母变为小写字母，而后判断字母j是否存在
		System.out.println(stream.filter((ele)->ele.toLowerCase().contains("j")).count());
	}
}
```

但是以上的程序只是实现了一些最基础的数据的个数的统计，而更多情况下我们可能需要的是获取里面的满足条件的数据内容，所以此时可以实现数据的采集操作。

范例：数据采集

```
package cn.mldn.demo;
```

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		List<String> all = new ArrayList<String>();
		Collections.addAll(all, "Java", "JavaScript", "Python", "Ruby", "Go");
		Stream<String> stream = all.stream(); // 可以获得Stream接口对象
		// 要求将每一个元素的字母变为小写字母，而后判断字母j是否存在，将满足条件的数据
收集起来转为List集合
		List<String> result = stream.filter((ele) ->
ele.toLowerCase().contains("j")).collect(Collectors.toList());
		System.out.println(result);
	}
}
```

在Stream数据流处理的过程之中还允许进行数据的分页操作，提供有两个方法：

·设置取出的最大数据量：public Stream<T> limit(long maxSize);

·跳过指定数据量：public Stream<T> skip(long n);

范例：观察分页

```
package cn.mldn.demo;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		List<String> all = new ArrayList<String>();
		Collections.addAll(all, "Java", "JavaScript", "JSP", "Json", "Python", "Ruby", "Go");
		Stream<String> stream = all.stream(); // 可以获得Stream接口对象
		// 要求将每一个元素的字母变为小写字母，而后判断字母j是否存在，将满足条件的数据
收集起来转为List集合
		List<String> result = stream.filter((ele) ->
ele.toLowerCase().contains("j")).skip(2).limit(2)
				.collect(Collectors.toList());
		System.out.println(result);
	}
}
```

Stream的操作主要是利用自身的特点实现数据的分析处理操作。

■MapReduce基础模型

在进行数据分析的处理之中有一个最重要的基础模型：MapReduce模型，对于这个模型一共是分为两个部分：Map处理部分、Reduce分析部分部分，在进行数据分析之前必须

要对数据进行合理的 处理，而后才可以做统计分析操作。

范例：MapReduce基础模型

```
package cn.mldn.demo;
import java.util.ArrayList;
import java.util.DoubleSummaryStatistics;
import java.util.List;
public class JavaAPIDemo {
	public static void main(String[] args) throws Exception {
		// 如果要想使用Stream进行分析处理，则一定要将全部要分析的数据保存在集合之中
		List<Order> all = new ArrayList<Order>();
		all.add(new Order("小强娃娃", 9.9, 10));
		all.add(new Order("林弱充气娃娃", 2987.9, 3));
		all.add(new Order("不强牌笔记本电脑", 8987.9, 8));
		all.add(new Order("弱强茶杯", 2.9, 800));
		all.add(new Order("阿强牌煎饼", 0.9, 138));
		// 分析购买商品之中带有“强”的信息数据，并且进行商品单价和数量的处理，随后分析汇总
		DoubleSummaryStatistics stat = all.stream().filter((ele) -> ele.getName().contains("强"))
				.mapToDouble((orderObject) -> orderObject.getPrice() * orderObject.getAmount()).summaryStatistics();
		System.out.println("购买数量：" + stat.getCount());
		System.out.println("购买总价：" + stat.getSum());
		System.out.println("平均花费：" + stat.getAverage());
		System.out.println("最高花费：" + stat.getMax());
		System.out.println("最低花费：" + stat.getMin());
	}
}
class Order {	// 订单信息
	private String name ; // 商品名称
	private double price ; // 商品单价
	private int amount ; // 商品数量
	public Order(String name,double price,int amount) {
		this.name = name ;
		this.price = price ;
		this.amount = amount ;
	}
	public int getAmount() {
		return amount;
	}
	public String getName() {
		return name;
	}
	public double getPrice() {
		return price;
	}
}
```

这些分析操作只是JDK本身提供的支持，而实际之中，肯定不可能这样进行，因为所有的数据如果都保存在内存中，那么面对于大数据的环境，就溢出了。