Guava学习之RangeSet

前面谈到了Guava中新引入的Range类，也了解了其中的作用，那么今天来谈谈Guava中用到Range来的地方：RangeSet类。RangeSet类是用来存储一些不为空的也不相交的范围的数据结构。假如需要向RangeSet的对象中加入一个新的范围，那么任何相交的部分都会被合并起来，所有的空范围都会被忽略。

讲了这么多，我们该怎么样利用RangeSet？RangeSet类是一个接口，需要用它的子类来声明一个RangeSet型的对象，实现了RangeSet接口的类有ImmutableRangeSet和TreeRangeSet，ImmutableRangeSet是一个不可修改的RangeSet，而TreeRangeSet是利用树的形式来实现。下面主要谈TreeRangeSet的用法:

```
import com.google.common.collect.*;

/**
 * User: 过往记忆
 * Email:wyphao.2007@163.com
 * Blog:
 * Date: 13-7-17
 * Time: 下午4:10
 */

public void testRangeSet(){
        RangeSet rangeSet = TreeRangeSet.create();
        rangeSet.add(Range.closed(1, 10));
        System.out.println(rangeSet);
        rangeSet.add(Range.closedOpen(11, 15));
        System.out.println(rangeSet);
        rangeSet.add(Range.open(15, 20));
        System.out.println(rangeSet);
        rangeSet.add(Range.openClosed(0, 0));
        System.out.println(rangeSet);
        rangeSet.remove(Range.open(5, 10));
        System.out.println(rangeSet);
    }
```

上面函数的运行结果如下所示：

```
{[1..10]}
{[1..10][11..15)}
{[1..10][11..15)(15..20)}
```

{[1..10][11..15)(15..20)}
{[1..5][10..10][11..15)(15..20)}

对Range类的方法不熟悉，请阅读《Guava学习之Range》。

注意：RangeSet需要充分利用JDK 1.6中NavigableMap特性，所以JDK1.6以下版本无法使用。

那如果我们需要遍历rangeSet中的所有元素可以用下面方法实现

```
public void iteratorRangeSet(RangeSet integerRangeSet) {
    if(integerRangeSet == null){
        return;
    }

    Set<Range> ranges = integerRangeSet.asRanges();
    Iterator<Range> iterator = ranges.iterator();
    while(iterator.hasNext()){
        Range next = iterator.next();
        System.out.println(next);
    }
}
```

运行结果：

```
[1..5]
[10..10]
[11..15)
(15..20)
```

如果我们需要得到rangeSet互补的范围，我们可以用RangeSet提供的complement()方法，rangeSet.complement()同样是一个RangeSet，其中的元素也是互不相交、且不为空的RangeSet，那么rangeSet的互补集可以像下面这样来写：

```
RangeSet complement = rangeSet.complement();
System.out.println(complement);
```

得到的结果是：

{(-∞..1)(5..10)(10..11)[15..15][20..+∞)}

正好是rangeSet的互补。
如果需要在rangeSet中查询某个元素是否在rangeSet中，可以用contains(C)来实现，其中C extends java.lang.Comparable。比如我想得到上述rangeSet是否包含15，可以这样写：

```
boolean isIn = rangeSet.contains(15);
System.out.println(isIn);//false，因为上述范围不包含元素15.
```

如果想知道某个元素是在rangeSet中哪个范围里面，可以这样写：

```
Range integerRange = rangeSet.rangeContaining(17);
System.out.println(integerRange);
//输出(15..20)，因为17被包含在(15..20)中，所以输出这个范围。
```

如果想知道某个范围是否包含在rangeSet的范围中，可以这样写：

```
boolean encloses = rangeSet.encloses(Range.closedOpen(18, 20));
System.out.println(encloses);//true.因为范围(18,20)包含在范围(15,20)中
encloses = rangeSet.encloses(Range.closedOpen(5, 20));
System.out.println(encloses);//false.因为范围(5,20)不被rangeSet中任何范围包含.
```

（完）