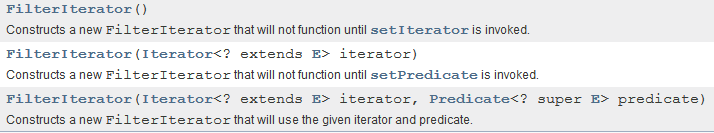
FilterIterator<E>过滤迭代器

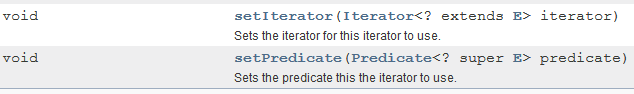
1. **FilterIterator<E>**存在于org.apache.commons.collections4.iterators包中，直接继承于java.lang.Object类，直接实现了java.util.Iterator接口，有一个直接子类[UniqueFilterIterator](mk:@MSITStore:E:\API\Commons-Collections4-4.1.CHM::/org/apache/commons/collections4/iterators/UniqueFilterIterator.html)（重复过滤迭代器）。commons给的过滤迭代器很少，主要是创建过滤器较方便，只需要创建一个Predicate类，将Predicate对象传入FilterIterator即可。
2. **FilterIterator<E>类：**

**（1）构造方法**：

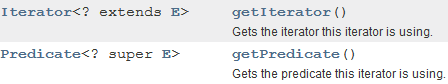


（2）成员方法：

①set参数：



②get参数：



③hasNext方法：。

④remove方法：。

⑤next方法：。

1. **UniqueFilterIterator<E>**类：**重复过滤迭代器**。存在于包org.apache.commons.collections4.iterators中，继承于FilterIterator类。需要传入一个迭代器iterator，然后对iterator进行包装，添加新的功能，用于过滤掉**相同的元素，生成一个新的没有重复元素的迭代器。**

**构造方法：参数就是一个迭代器。**



**其成员方法都是继承于FilterIterator。**

**示例：**

List<String> list = **new** ArrayList<String>();

list.add("abc");

list.add("abc");

list.add("def");

Iterator<String> iterator = list.iterator();

UniqueFilterIterator<String> ufi = **new** UniqueFilterIterator<>(iterator);

**while**(ufi.hasNext()) {

System.***out***.println(ufi.next());

}

输出结果：

1. **自定义过滤迭代器**：**FilterIterator = Iterator+Predicate**。只需要自定义一个**Predicate类**，将Predicate对象传入FilterIterator即可。

**回文过滤迭代器：**

List<String> list = **new** ArrayList<String>();

list.add("refer");

list.add("mom");

list.add("dad");

list.add("shandong");

Iterator<String> iterator = list.iterator();

Predicate<String> predicate = **new** Predicate<String>() {

@Override

**public** **boolean** evaluate(String str) {

**return** **new** StringBuilder(str).reverse().toString().equals(str);//回文判断

}

};

FilterIterator<String> filterIterator = **new** FilterIterator<>(iterator, predicate);

**while**(filterIterator.hasNext()) {

System.***out***.println(filterIterator.next());

}

结果：