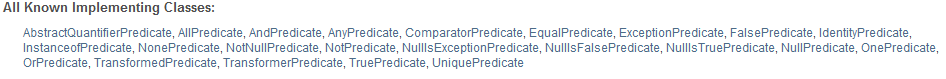
Predicate接口及其实现类和PredicateUtils工具类

1. **Predicate**是一个**接口**,存在 **org.apache.commons.collections4包中**。有一个子接口***PredicateDecorator***，有许多**实现子类**：

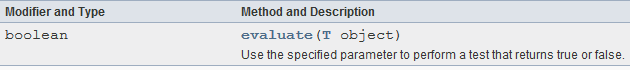


接口Predicate定义在包**org.apache.commons.collections4中，而此接口的一些实现类定义在了**org.apache.commons.collections4.functors包中。

**接口Transformer也是如此**。**org.apache.commons.collections4包中只定义接口和XxxxUtils工具类，对于其他对接口的实现类都定义在了子包中。**

Predicate是一种函数式编程，断言，封装条件或判别式，if…else替换。

只定义了一个方法：**evaluate（Object object），返回布尔值。**



**自定义Predicate实现子类**：只要是实现了evaluate这个抽象方法，即可自定义一个Predicate实现子类。

1. **Predicate接口**的实现子类介绍：

**共性：都是final类，都存在于包**org.apache.commons.collections4.functors中，都可以通过**静态方法**获得对象，都对Predicate接口的evaluate方法进行了实现。都可以通过PredicateUtils工具类中的静态方法获取相应的实现类的对象。

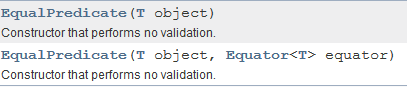
**如：**



1. **EqualPredicate：相等的判断。final类，存在于包**org.apache.commons.collections4.functors中。

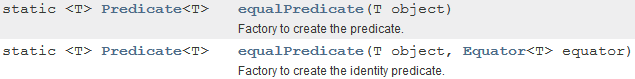
A）.创建对象有两种方法：

* 1. 通过构造方法，传入参数即可。



如：Predicate<String> predicate = **new** EqualPredicate<String>("maozhuxi");

* 1. 通过静态方法equalPredicate。注意：泛型不需要再次声明了，因为**参数的类型就是泛型的类型**。



如：Predicate<String> predicate = EqualPredicate.equalPredicate("maozhuxi");

B). 有两个成员方法：

* 1. **evaluate（T object）**：对接口Predicate<T>的实现。



②getValue（）:获取创建此对象时的object的值，即调用的就是object的**toString方法**。



1. NotNullPredicate：非空的判断和NullPredicate类。

都是final类，都存在于包org.apache.commons.collections4.functors中。

**创建对象**采用了单例模式，可以通过静态方法notNullPredicate（），构造方法私有化了。也可通过**静态属性INSTANCE**创建对象。



**不需要传入参数**，因为比较只是对单个判断是否非空，在执行方法时传入即可。因为一个NotNullPredicate就可以对任何对象进行判断是否非空，所以采用单例模式。



**方法：**

**对接口Predicate实现了方法evaluate（Object obj）。**

**如：**Predicate<String> predicate = NotNullPredicate.*notNullPredicate*();

Predicate<String> predicate3 = NotNullPredicate.***INSTANCE***;

Predicate<String> predicate2 = NullPredicate.*nullPredicate*();

Predicate<String> predicate4 = NullPredicate.***INSTANCE***;

1. UniquePredicate:唯一性判断。final类。

创建对象通过构造方法或静态方法方法创建。





实现Predicate接口中的evaluate方法:



示例：

Predicate<Integer> predicate = **new** UniquePredicate<Integer>();

Predicate<Integer> predicate2 = UniquePredicate.*uniquePredicate*();

List<Integer> list = PredicatedList.*predicatedList*(

**new** ArrayList<Integer>(), predicate);

list.add(343);

System.***out***.println("ok1");

list.add(322);

System.***out***.println("ok2");

list.add(322);

System.***out***.println("ok3");

1. **拓展**：类PredicateList:对List进行装饰，添加判断的功能。

通过静态方法predicatedList（list，predicate）获得对象。传入的List是java.util包中的List。



如传入一个List中的对象不能为null，则：

List<String> list = PredicatedList.predicatedList(new ArrayList<String>(),NotNullPredicate.INSTANCE);

list.add("奥巴马");

list.add(new String());

// String str = null;

// list.add(str);//抛出异常

如：传入的对象必须是唯一的：

List<String> list = PredicatedList.predicatedList(new ArrayList<String>(),UniquePredicate.getInstance());

list.add("aobama");

list.add("aobama");// 报出异常

1. 自定义**Predicate对象**，只需要实现其中的**evaluate（Object obj）**方法即可。

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

@Override

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

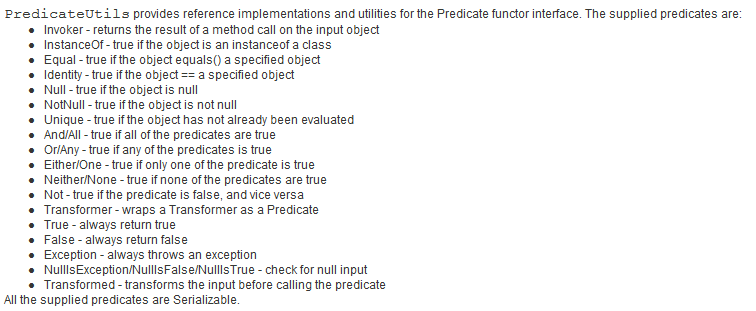
**return** object .startsWith("zhao");

}

};

System.***out***.println(predicate .evaluate("zhahongzxy"));;

1. **PredicateUtils类**是一个Predicate的工具类，存在于org.apache.commons.collections包中，直接继承与java.lang.Objec类，主要有两个功能，即用于**快速获取特定功能的Predicate对象**和**实现两个或多个Predicate对象之间的与或非等**。
2. 利用PredicateUtils工具类获取常用的Predicate对象：



1. equalPredicate：



1. identityPredicate



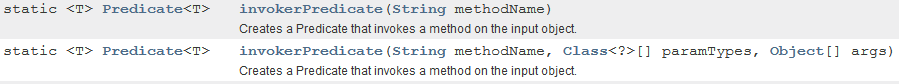
1. instanceofPredicate



1. uniquePredicate



1. invokePredicate



1. notNullPredicate



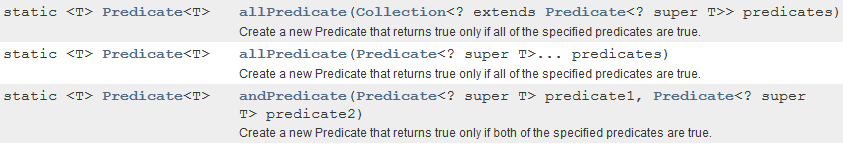
1. 等等。
2. 多个**Predicate对象**联合：

采用**PredicateUtils工具类**，里面的方法都是**静态方法**。

（1）**andPredicate将两个Predicate联合。联合后的Predicate对象，可以继续与其他Predicate对象联合，从而实现多个Predicate的联合。**



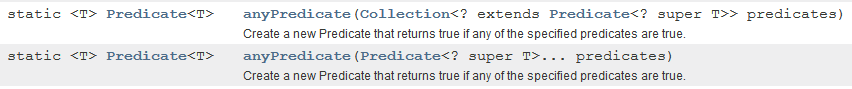
（2）**allPredicate方法**可以将多个**Predicate对象**直接联合成一个Predicate对象。多个Predicate可以存放于一个**集合**Collection中，也可以存放到一个**Predicate数组**中。



（3）**orPredicate或eitherPredicate或anyPredicate**







**orPredicate和eitherPredicate的区别**：

orPredicate：在两个中有任何一个满足或都满足返回true；

eitherPredicate：只有在两者中只有一个为true时，才返回true。

对于anyPredicate的使用方法同allPredicate，不再赘述。

示例：

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

@Override

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

**return** object.length() >= 6 && object.length() <= 16;

}

};

Predicate<String> notNullPredi = PredicateUtils.*notNullPredicate*();

Predicate<String> uniquePredi = PredicateUtils.*uniquePredicate*();

// (1) 利用andPredicatefangfa

// Predicate<String> predicates =

// PredicateUtils.andPredicate(selfPredicate, notNullPredi );

// predicates = PredicateUtils.andPredicate(predicates, uniquePredi);

// // (2) 利用数组联合

// Predicate[] predicates = new Predicate[] {

// selfPredicate, notNullPredi, uniquePredi };

// Predicate pres = PredicateUtils.allPredicate(predicates);

// (3)利用集合

List<Predicate<String>> presList = **new** ArrayList<Predicate<String>>();

presList.add(selfPredicate);

presList.add(notNullPredi);

presList.add(uniquePredi);

Predicate pres = PredicateUtils.*allPredicate*(presList);

// (4) 新特性，参数可以为任意个，直接传入多个Predicate对象即可。

Predicate<String> predicates = PredicateUtils.*allPredicate*(

selfPredicate, notNullPredi, uniquePredi);

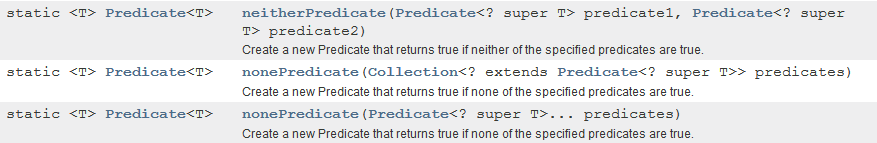
List<String> list = PredicatedList.*predicatedList*(

**new** ArrayList<String>(), predicates);

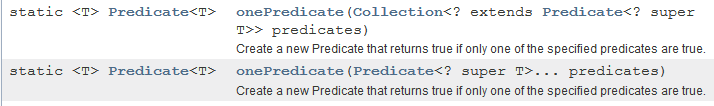
// list集合需要满足字符串大小在6到16之间，且非空，且唯一。

1. PredicateUtils中的其他方法：
2. **neitherPredicate或NonePredicate**

同eitherPredicate或anyPredicate使用。



1. onePredicate：仅当只有一个为true时，才返回true。



1. 等等。