链滴

Guava 中针对集合的 filter 和过滤功能

作者：jsy

原文链接：https://ld246.com/article/1493451481279

来源网站：链滴

```
@Test
public  void whenFilterWithIterables_thenFiltered() {
 List<String> names = Lists.newArrayList("John", "Jane", "Adam", "Tom");
 Iterable<String> result = Iterables.filter(names, Predicates.containsPattern("a"));

 assertThat(result, containsInAnyOrder("Jane", "Adam"));
}
@Test
public  void whenFilterWithCollections2_thenFiltered() {
 List<String> names = Lists.newArrayList("John", "Jane", "Adam", "Tom");
 Collection<String> result = Collections2.filter(names, Predicates.containsPattern("a"));

 assertEquals(2, result.size());
 assertThat(result, containsInAnyOrder("Jane", "Adam"));

 result.add("anna");
 assertEquals(5, names.size());
}
@Test
public  void whenFilterCollectionWithCustomPredicate_thenFiltered() {
 Predicate<String> predicate =  new Predicate<String>() {
 @Override
 public boolean apply(String input) {
 return input.startsWith("A") || input.startsWith("J");
 }
 };
```

```
 List<String> names = Lists.newArrayList("John", "Jane", "Adam", "Tom");
 Collection<String> result = Collections2.filter(names, predicate);

 assertEquals(3, result.size());
 assertThat(result, containsInAnyOrder("John", "Jane", "Adam"));
}
@Test
public  void whenTransformWithIterables_thenTransformed() {
 Function<String, Integer> function =  new Function<String, Integer>() {
 @Override
 public Integer apply(String input) {
 return input.length();
 }
 };

 List<String> names = Lists.newArrayList("John", "Jane", "Adam", "Tom");
 Iterable<Integer> result = Iterables.transform(names, function);

 assertThat(result, contains(4, 4, 4, 3));
}
@Test
public  void whenTransformingUsingComposedFunction_thenTransformed() {
 Function<String,Integer> f1 =  new Function<String,Integer>(){
 @Override
 public Integer apply(String input) {
 return input.length();
 }
```

```
};

Function<Integer,Boolean> f2 =  new Function<Integer,Boolean>(){
@Override
public Boolean apply(Integer input) {
return input %  2  ==  0;
}
};

List<String> names = Lists.newArrayList("John", "Jane", "Adam", "Tom");
Collection<Boolean> result = Collections2.transform(names, Functions.compose(f2, f1));

assertEquals(4, result.size());
assertThat(result, contains(true, true, true, false));
}
@Test
public  void whenFilteringAndTransformingCollection_thenCorrect() {
Predicate<String> predicate =  new Predicate<String>() {
@Override
public boolean apply(String input) {
return input.startsWith("A") || input.startsWith("T");
}
};

Function<String, Integer> func =  new Function<String,Integer>(){
@Override
public Integer apply(String input) {
```

```
return input.length();
}
};

List<String> names = Lists.newArrayList("John", "Jane", "Adam", "Tom");
Collection<Integer> result = FluentIterable.from(names)
.filter(predicate)
.transform(func)
.toList();

assertEquals(2, result.size());
assertThat(result, containsInAnyOrder(4, 3));
}
```