Recipe 0.6. Creating a generic condition

Table of contents

1 Problem.....2
2 Definitions.....2
3 Solution.....2
4 Discussion.....3
5 See Also.....3

1. Problem

You want to create a condition on any type.

2. Definitions

```
Conditional or boolean expressions are expressions that are
either true or false.
```

In the JPL, there are a list of operators for testing if a condition is true. There are operators for equality, inequality, membership (contains, belongs to a set or a domain of definition) ...

3. Solution

Here is an example of code to define a specific condition:

```
// a condition on doubles
Condition<Double> cond =
        new ConditionImpl.Builder<Double, Double>(54.3).operator(
                OperatorGreaterThan.newInstance()).build();

// a list of doubles to test
List<Double> l =
        Arrays.asList(1.9, 4.3, 56.2, 56.1, 120., 23., 45., 89.4);

// counter
int count = 0;
for (double d : l) {
        if (cond.isTrue(d)) {
                count++;
        }
}
Assert.assertEquals(4, count);
```

Sometimes, the object to test is not of the same type than the operand to test with. In this case, we will have to define how to find it to the builder:

```
import import org.apache.commons.collections15.Transformer;

// a list of doubles
List<Double> l = Arrays.asList(32., 33.0, 34.0);

// a stub method to reach a Double from a list of Doubles
Transformer<List<Double>, Double> getTheFirstElt =
        new Transformer<List<Double>, Double>() {

        public Double transform(final List<Double> l) {
```

```
                return l.get(0);
        }
};

// here is the condition that test list of doubles against the double 32.0
// the operator by default is "equals"
ConditionImpl<List<Double>, Double> cond =
        new ConditionImpl.Builder<List<Double>, Double>(32.0)
                .accessor(getTheFirstElt)
                .build();

Assert.assertTrue(cond.isTrue(l));
```

4. Discussion

5. See Also

See also how to create conditional expression.