LINGI2251

Software Quality Assurance

Assignment 1

Vincent Tessier - Nathan Rullier - Glazoom

Spring 2020

**Functional testing**

1. the list of categories and choices (or the PICT model file) with explanations; statistics on the generated test cases.

Parameters: Template, MacroResolver

Template Categories and choices

* Input string
  + Null [single]
  + Valid (Length > 0)
* MacroPrefix
  + Null (StrictFormat)
  + Valid
* MacroStart
  + Null [error]
  + Valid
* MacroEnd
  + Null [single]
  + Valid
* ResolveEscapes
  + True
  + False
* Number of escapes in the string
  + 0
  + [1,infinity[
* Missing key replacement
  + True
  + False
* Missing replacement key value
  + Null [single]
  + Valid
* ParseValues
  + True
  + False [single]

MacroResolver

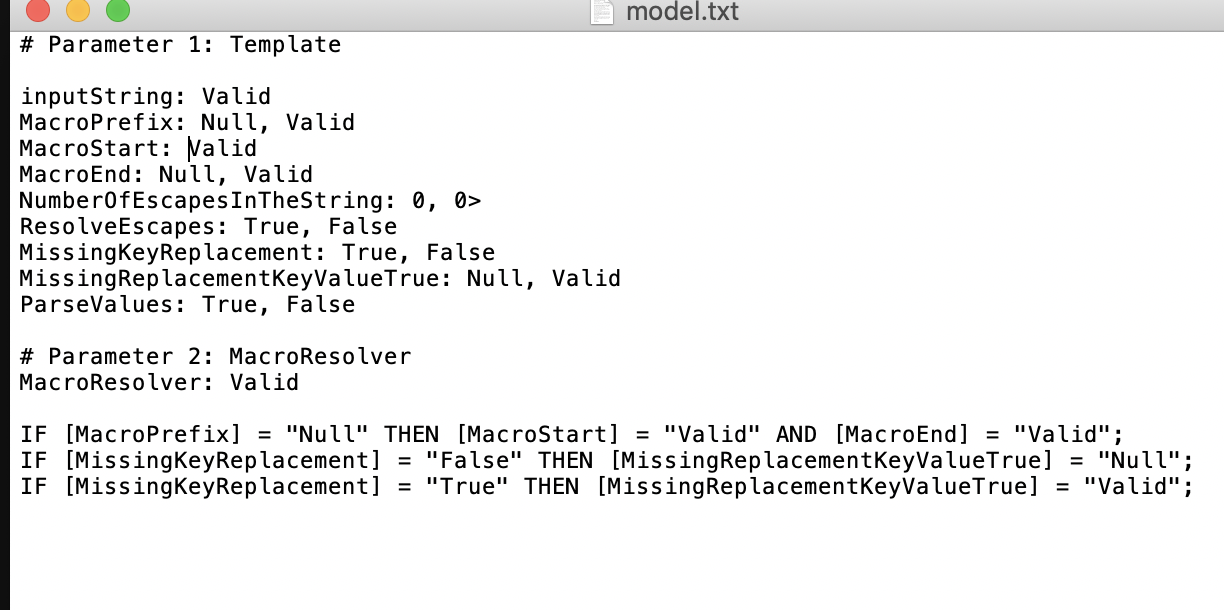
* String function
  + Valid
  + Null [error]

Constraint:

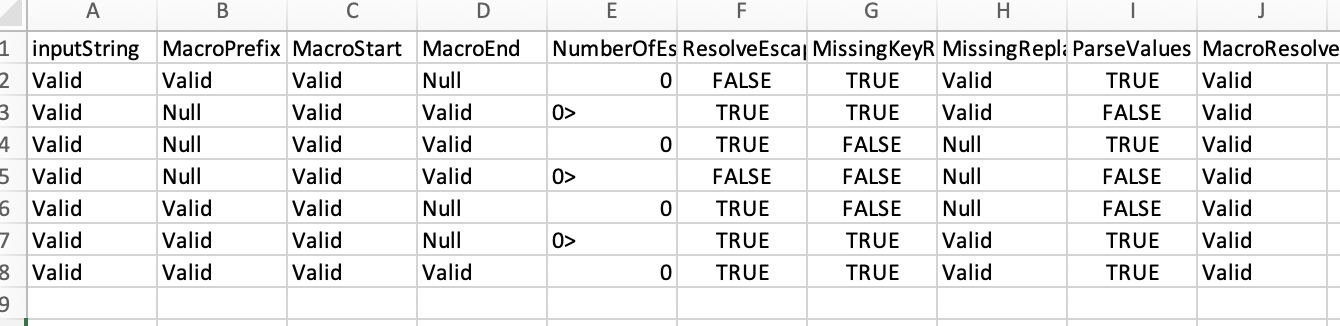
IF [MacroPrefix] = "Null" THEN [MacroStart] = "Valid" AND [MacroEnd] = "Valid";

//If the macro prefix is null, it means the format is strict and MacroStart and MacroEnd need to be valid or there will be an error.

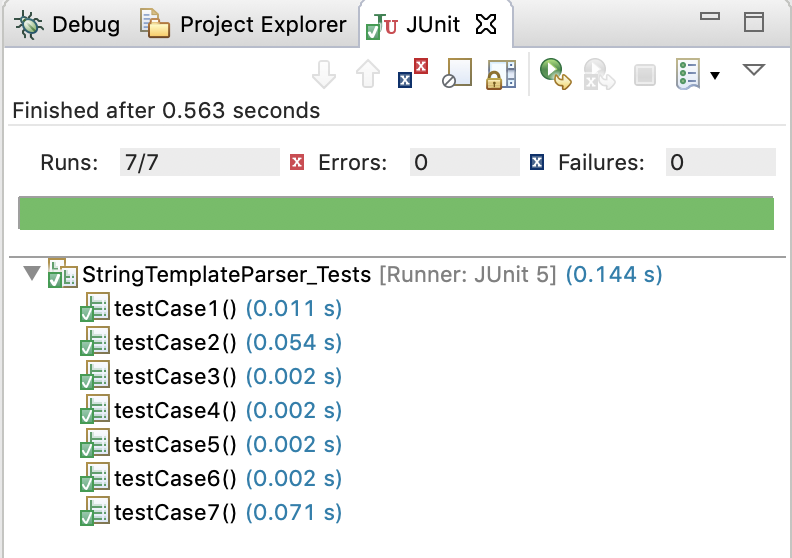
Pict input:



Output:



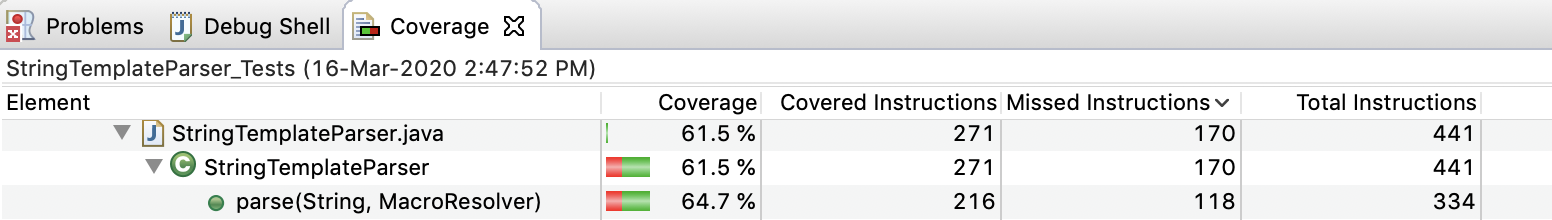
Tests results:



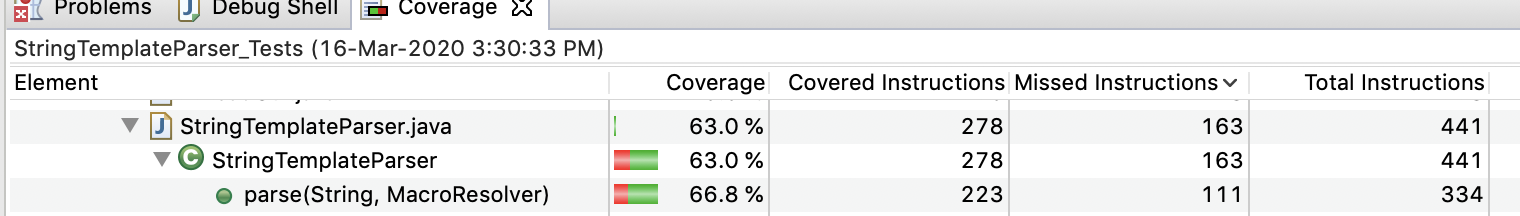
**Structural Testing**

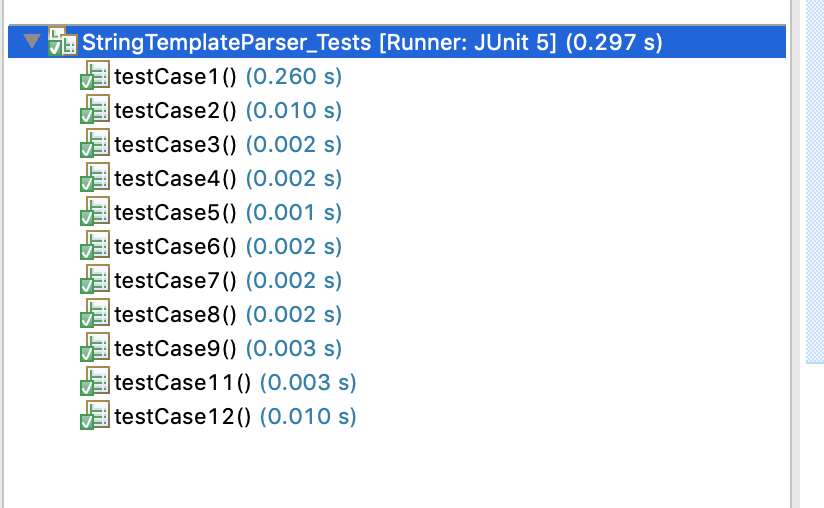
Coverage statistics

Initial coverage statistics



Completed test suite

test cases



**Fault-Based Testing**

1. coverage statistics with initial and with completed test suite; discussion of unkillable mutants; added test cases.

**Appendix**

package jodd.util;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.Assertions;

class StringTemplateParser\_Tests {

// 7 test cases of the PICT output file (valid tests)

@Test

void testCase1() { //test1 Missing replacement key value = true

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

String inputString = "xxx${small}xxx";

stringTemplateParserTest.setMacroPrefix("$");

stringTemplateParserTest.setMacroStart("${");

stringTemplateParserTest.setMacroEnd("");

stringTemplateParserTest.setResolveEscapes(false);

stringTemplateParserTest.setReplaceMissingKey(true);

stringTemplateParserTest.setMissingKeyReplacement("%");

stringTemplateParserTest.setParseValues(true);

assertEquals("xxxsmall}xxx",stringTemplateParserTest.parse(inputString, String::toUpperCase), "test1 ");

}

@Test

void testCase2() {

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

String inputString = "xxx${sm all}xxx";

stringTemplateParserTest.setMacroPrefix("");

stringTemplateParserTest.setMacroStart("${");

stringTemplateParserTest.setMacroEnd("}");

stringTemplateParserTest.setResolveEscapes(true);

stringTemplateParserTest.setReplaceMissingKey(true);

stringTemplateParserTest.setMissingKeyReplacement("%");

stringTemplateParserTest.setParseValues(false);

assertEquals("XXXSM ALLXXX",stringTemplateParserTest.parse(inputString, String::toUpperCase), "test2 ");

}

@Test

void testCase3() {

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

String inputString = "xxx${small}xxx";

stringTemplateParserTest.setMacroPrefix("");

stringTemplateParserTest.setMacroStart("${");

stringTemplateParserTest.setMacroEnd("}");

stringTemplateParserTest.setResolveEscapes(true);

stringTemplateParserTest.setReplaceMissingKey(false);

stringTemplateParserTest.setMissingKeyReplacement("");

stringTemplateParserTest.setParseValues(true);

assertEquals("XXXSMALLXXX",stringTemplateParserTest.parse(inputString, String::toUpperCase), "test3 ");

}

@Test

void testCase4() {

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

String inputString = "xxx${sm all}xxx";

stringTemplateParserTest.setMacroPrefix("");

stringTemplateParserTest.setMacroStart("${");

stringTemplateParserTest.setMacroEnd("}");

stringTemplateParserTest.setResolveEscapes(false);

stringTemplateParserTest.setReplaceMissingKey(false);

stringTemplateParserTest.setMissingKeyReplacement("");

stringTemplateParserTest.setParseValues(false);

assertEquals("XXXSM ALLXXX",stringTemplateParserTest.parse(inputString, String::toUpperCase), "test4 ");

}

@Test

void testCase5() {

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

String inputString = "xxx${small}xxx";

stringTemplateParserTest.setMacroPrefix("$");

stringTemplateParserTest.setMacroStart("${");

stringTemplateParserTest.setMacroEnd("}");

stringTemplateParserTest.setResolveEscapes(true);

stringTemplateParserTest.setReplaceMissingKey(false);

stringTemplateParserTest.setMissingKeyReplacement("");

stringTemplateParserTest.setParseValues(false);

assertEquals("xxxSMALLxxx",stringTemplateParserTest.parse(inputString, String::toUpperCase), "test5 ");

}

@Test

void testCase6() {

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

String inputString = "xxx${sm all}xxx";

stringTemplateParserTest.setMacroPrefix("$");

stringTemplateParserTest.setMacroStart("${");

stringTemplateParserTest.setMacroEnd("");

stringTemplateParserTest.setResolveEscapes(true);

stringTemplateParserTest.setReplaceMissingKey(true);

stringTemplateParserTest.setMissingKeyReplacement("%");

stringTemplateParserTest.setParseValues(true);

assertEquals("xxxsm all}xxx",stringTemplateParserTest.parse(inputString, String::toUpperCase), "test6 ");

}

@Test

void testCase7() {

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

String inputString = "xxx${small}xxx";

stringTemplateParserTest.setMacroPrefix("$");

stringTemplateParserTest.setMacroStart("${");

stringTemplateParserTest.setMacroEnd("}");

stringTemplateParserTest.setResolveEscapes(true);

stringTemplateParserTest.setReplaceMissingKey(true);

stringTemplateParserTest.setMissingKeyReplacement("%");

stringTemplateParserTest.setParseValues(true);

assertEquals("xxxSMALLxxx",stringTemplateParserTest.parse(inputString, String::toUpperCase), "test7");

}

//informative tests that gives error or with a Single constraint

@Test //test8 input string = null [Single]

void testCase8() {

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

stringTemplateParserTest.setReplaceMissingKey(true);

stringTemplateParserTest.setMissingKeyReplacement("");

stringTemplateParserTest.setResolveEscapes(true);

stringTemplateParserTest.setMacroPrefix("$");

stringTemplateParserTest.setMacroStart("${");

stringTemplateParserTest.setMacroEnd("}");

stringTemplateParserTest.setParseValues(true);

assertEquals("",stringTemplateParserTest.parse("", String::toUpperCase), "test2: input string = null ...");

}

@Test //test9 String function = null

void testCase9() {

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

stringTemplateParserTest.setResolveEscapes(false);

stringTemplateParserTest.setMacroPrefix("$");

stringTemplateParserTest.setMacroStart("${");

stringTemplateParserTest.setMacroEnd("}");

stringTemplateParserTest.setParseValues(false);

Assertions.assertThrows(NullPointerException.class, () -> {

stringTemplateParserTest.parse("xxx${small}xxx", null);

});

}

@Test //test10 MacroStart = null (stock in a infinite while loop, there's a problem here.

void testCase10() {

StringTemplateParser stringTemplateParserTest = new StringTemplateParser();

stringTemplateParserTest.setReplaceMissingKey(true);

stringTemplateParserTest.setMissingKeyReplacement("");

stringTemplateParserTest.setResolveEscapes(true);

stringTemplateParserTest.setMacroPrefix("${");

stringTemplateParserTest.setMacroStart("");

stringTemplateParserTest.setMacroEnd("}");

stringTemplateParserTest.setParseValues(true);

assertEquals("",stringTemplateParserTest.parse("xxx${small}xxx", String::toUpperCase), "test3 prefix = null, start = valid, end = valid");

}