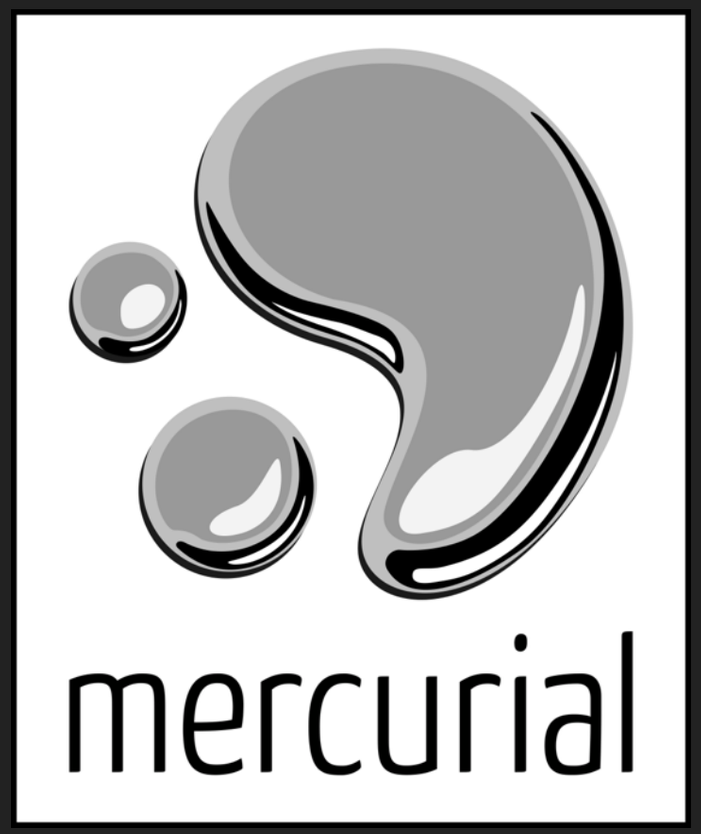

CHRIS SIGMUND | JESSE HUNT | KYLE BROOKS

MERCURIAL

INTRODUCTIONS

THE PROJECT

MERCURIAL

WHAT IS MERCURIAL

- Version Control System (Think Git)
- Written in Python
- Cross-Platform Compatible

TEAM GOALS

- Create method tests for Mercurial classes
- Develop a test automation script
- Conform to a testing standard

FRAMEWORK

TEST AUTOMATION

- All relevant files stored in Test Automation folder/subfolders
- Test Automation Directory
  - Oracles
  - Project
  - Reports
  - Scripts
  - testCases

TESTAUTOMATION/TESTCASES

- Comprised only of .txt files
- Text file for each individual test case

testCase01.txt

```
01
Break a given integer (representing time in seconds) into largest units of time
possible
progress
fmtremaining
100
1m40s
```

TESTAUTOMATION/SCRIPTS

- Contains the master script, runAllTests.py
- Executes all tests in testCases Directory
- Outputs directly to console
- Formats results into .html file

```
def runTests(report, count):

        passCount = 0 #number of passed tests
        failCount = 0 #numer of failed tests

        for infile in listing:
                try:
                        with open(testCasePath + '/' + infile, 'r') as f:
                                testList = f.read().splitlines()
                                iden = testList[0]
                                req = testList[1]
                                component = testList[2]
                                method = testList[3]
                                inp = testList[4]
                                outcome = testList[5]
                                compAndMethod = component + "." + method + "()"

                                print "ID: ", iden


                                ##Format: TestID, Component/Method, Input, Expected Output, Actual Output, Pass/Fail
                                passFormat = '<tr bgcolor="#00FF00"><td>{}</td><td>{}</td><td>{}</td><td>{}</td><td>{}</td><td>{}</td></tr>'
                                failFormat = '<tr bgcolor ="#FF0000"><td>{}</td><td>{}</td><td>{}</td><td>{}</td><td>{}</td><td>{}</td></tr>'


                                #Execute test file script
                                import_module = import_base + component
                                exec( import_module )
                                print "Requirements: ", req
                                print "Component: ", component
                                print "Method: ", method
                                print "Input: ", inp
                                print "Expected Outcome: ", outcome


                                statement = component + "." + method+  "(" + inp + ")"
                                result = eval(statement)
                                print "Result: " + str(result)



                                if (str(result) == str(outcome)):
                                        passCount += 1
                                        passFormat = passFormat.format(iden, compAndMethod, inp, outcome, str(result), "PASS")
                                        report.write(passFormat)
                                        print("PASS\n")
                                else:
                                        failCount += 1
                                        failFormat = failFormat.format(iden, compAndMethod, inp, outcome, str(result), "FAIL")
                                        report.write(failFormat)
                                        print("FAIL\n")
        except:
                failCount += 1
                failFormat = failFormat.format(iden, compAndMethod, inp, outcome, "ERROR", "FAIL")
                report.write(failFormat)
                e = sys.exc_info()[0]
```

TESTAUTOMATION/REPORTS

- Contains the results of all test runs, stored in report.html
- report.html is overwritten each time runAllTests.py is executed

Test ID	Component.method()	Input	Expected Output	Actual Output	Pass/Fail
01	progress.fmtremaining()	100	1m40s	1m40s	PASS
02	simplemerge.intersect()	(0,100), (50,150)	(50, 100)	(50, 100)	PASS
03	worker.countcpus()		4	4	PASS
04	templatefilters.count()	"abcde"	5	5	PASS
05	namespaces.tolist()	"1" "2" "3" "4"	['1234']	['1234']	PASS
06	progress.fmtremaining()	59	59s	59s	PASS
07	progress.fmtremaining()	0	00s	00s	PASS
08	progress.fmtremaining()	3601	1h01m	1h01m	PASS
09	progress.fmtremaining()	3599	59m59s	59m59s	PASS
10	progress.fmtremaining()	-59	-59s	-59s	PASS
11	simplemerge.intersect()	(0,10),(5,15)	(5, 10)	(5, 10)	PASS
12	simplemerge.intersect()	(0,100), (50,50)	None	None	PASS
13	simplemerge.intersect()	(0,10), (10,10)	None	None	PASS
14	simplemerge.intersect()	(0,2), (1,15)	(1, 2)	(1, 2)	PASS
15	simplemerge.intersect()	(0,10), (10,15)	None	None	PASS
16	templatefilters.count()	""	0	0	PASS
17	templatefilters.count()	" n "	5	5	PASS
18	templatefilters.count()	"1_%YTb"	6	6	PASS
19	templatefilters.count()	'aaaaaaaaaaaaaaa'	15	15	PASS
20	templatefilters.count()	"hello" "world"	10	10	PASS
21	namespaces.tolist()	"add" "These" "Words" "Together"	['addTheseWordsTogether']	['addTheseWordsTogether']	PASS
22	namespaces.tolist()	"This"" ""is"" ""a"" ""sentence."	['This is a sentence.']	['This is a sentence.']	PASS
23	namespaces.tolist()	"1""+""2""=""3"	['1+2=3']	['1+2=3']	PASS
24	namespaces.tolist()	" " " " "A" " " " "	[' A ']	[' A ']	PASS
25	namespaces.tolist()	""	['']	['']	PASS

Number of Tests: 25
Passed: 25
Failed: 0

MODIFYING MERCURIAL

GOAL

- EDIT TESTED METHODS IN MERCURIAL
- CHECK ROBUSTNESS OF OUR TEST CASES

METHODS CHANGED

- fmtremaining(int seconds)
- countcpus()
- intersect(ra, rb)
- count()
- tolist()

```
def intersect (ra, rb)

    #METHOD CODE HERE...

    #Original Code
    # return sa, sb

    ### MODIFIED CODE HERE
    return sb, sa
    ### MODIFIED CODE HERE

def fmtremaining(seconds):

  #Original code
  #if seconds < 60

  #### MODIFIED CODE HERE
  if seconds > 0 and seconds < 60:
  ### MODIFIED CODE HERE

  #REST OF METHOD CODE...

def count(i):

    #METHOD CODE HERE...

    #Original Code
    #return len(i)

    ### MODIFIED CODE
    return len(i) + 1

def tolist(val):

    #METHOD CODE HERE...

    if val is None:
        #Original code
        #return []

        ### MODIFIED CODE
        return 'Empty List'
        ### MODIFIED CODE
    else:
        return [val]
```

file:///home/christophersigmund/mylist1/RedTeam/TestAutomation/reports/report.html

Test ID	Component.method()	Requirements	Input	Expected Output	Actual Output	Pass/Fail
01	progress.fmtremaining()	Break a given integer (representing time in seconds) into largest units of time possible	100	1m40s	1m40s	PASS
02	simplemerge.intersect()	Given the range of two points, calculate and return the point of overlap between them	(0,100), (50,150)	(50, 100)	(100, 50)	FAIL
03	worker.countcpus()	Returns the number of processors available to the operating system		1	2	FAIL
04	templatefilters.count()	Return the length of the given string	"abcde"	5	6	FAIL
05	namespaces.tolist()	Return elements in given array concatenated into one element in a larger array	"1" "2" "3" "4"	['1234']	['1234']	PASS
06	progress.fmtremaining()	Break a given integer (representing time in seconds) into largest units of time possible	59	59s	59s	PASS
07	progress.fmtremaining()	Break a given integer (representing time in seconds) into largest units of time possible	0	00s	0m00s	FAIL
08	progress.fmtremaining()	Break a given integer (representing time in seconds) into largest units of time possible	3601	1h01m	1h01m	PASS
09	progress.fmtremaining()	Break a given integer (representing time in seconds) into largest units of time possible	3599	59m59s	59m59s	PASS
10	progress.fmtremaining()	Break a given integer (representing time in seconds) into largest units of time possible	-59	-59s	-1m01s	FAIL
11	simplemerge.intersect()	Given the range of two points, calculate and return the point of overlap between them	(0,10),(5,15)	(5, 10)	(10, 5)	FAIL
12	simplemerge.intersect()	Given the range of two points, calculate and return the point of overlap between them	(0,100), (50,50)	None	None	PASS
13	simplemerge.intersect()	Given the range of two points, calculate and return the point of overlap between them	(0,10), (10,10)	None	None	PASS
14	simplemerge.intersect()	Given the range of two points, calculate and return the point of overlap between them	(0,2), (1,15)	(1, 2)	(2, 1)	FAIL
15	simplemerge.intersect()	Given the range of two points, calculate and return the point of overlap between them	(0,10), (10,15)	None	None	PASS
16	templatefilters.count()	Return the length of the given string	""	0	1	FAIL
17	templatefilters.count()	Return the length of the given string	" n "	5	6	FAIL
18	templatefilters.count()	Return the length of the given string	"1_%YTb"	6	7	FAIL
19	templatefilters.count()	Return the length of the given string	'aaaaaaaaaaaaaaa'	15	16	FAIL
20	templatefilters.count()	Return the length of the given string	"hello" "world"	10	11	FAIL
21	namespaces.tolist()	Return elements in given array concatenated into one element in a larger arrays	"add" "These" "Words" "Together"	['addTheseWordsTogether']	['addTheseWordsTogether']	PASS
22	namespaces.tolist()	Return elements in given array concatenated into one element in a larger array	"This"" ""is"" ""a"" ""sentence."	['This is a sentence.']	['This is a sentence.']	PASS
23	namespaces.tolist()	Return elements in given array concatenated into one element in a larger array	"1""+""2""=""3"	['1+2=3']	['1+2=3']	PASS
24	namespaces.tolist()	Return elements in given array concatenated into one element in a larger array	None	[]	Empty List	FAIL
25	namespaces.tolist()	Return elements in given array concatenated into one element in a larger array	""	['']	['']	PASS

Number of Tests: 25
Passed: 12
Failed: 13

LESSONS LEARNED

LESSONS LEARNED

- How to create a testing framework
- Importance of keeping a schedule
- Thoroughly understanding the project
- Regular pushing to Git