Automated Testing Framework for Beets

By: Team Isn’tThisFun

What is Beets?

- Catalogs a music collection.
- Automatically improves its metadata by using MusicBrainz database.
- Provides a bouquet of tools for manipulating and accessing music.

.

Team's Goal

- To plan and implement an automated testing framework for Beets.
- To test five different methods and inject faults into the Beets source code to determine whether the framework catches bugs properly.

Deliverables & Acceptance Criteria

- Team created a script to run five automated tests on five methods within the project Beets, for a total of 25 tests.
- Tests compared the output of each method with an oracle for that method.
- Script displays the results in a web browser by running executable files, which create output files to be compared to the corresponding oracle files.
- The results of this comparison between the output file and oracle file determines whether the test passes and is displayed on an HTML page.

Screenshots of Testing Framework

Script

```
def main():
    readFiles()

def readFiles():
    """Reads in the lines from the test case specification file, passes them to
    parse, passes parsed list to driver specified by the test case specification
    file, and then calls the report function to handle building and displaying the
    results of the test in an HTML page."""

def parseFiles(inLineArray):
    """Parse the list of lines from the test case specification file to put these
    lines in the correct format."""

def compare(oracle, actualOutput):
    """Test whether the expected value from the test case specification file equals
    the actual output from the function."""

def strToFile(text, filename):
    """Write a file with the given name and the given text."""

def browseLocal(webpageText, filename='tempBrowseLocal.html'):
    """Start your webbrowser on a local file containing the text
    with given filename."""

def report(returnVal, contents2, outputVal):
    """ Write results of test to HTML file """
```

Driver

```
def driverDefaultFunc(info):
    """Calls the function specified in the test case specification file with the specified inputs,
    then returns the output."""

    inFuncName = info[3]
    inInputVal = info[4]

    ## If there is only one input value, call the function with one argument
    if (len(inInputVal) == 1):
        try:
            output = getattr(ui, inFuncName)(inInputVal[0])
        except TypeError as e:
            output = "TypeError"
        # except InputError:
        #     output = "InputError"
        except ValueError as e:
            output = "ValueError"
        except Exception, e:
            output = "Error"
    ## If there are two input values, call function with two arguments
    elif (len(inInputVal) == 2):
        try:
            output = getattr(ui, inFuncName)(inInputVal[0], inInputVal[1])
        except TypeError as e:
            output = "TypeError"
        except AssertionError as e:
            output = "AssertionError"
        except Exception, e:
            output = "Error"
    else:
        output = "Some Error"

    return output
```

HTML Report

Test #	Req. Tested	Component Tested	Method Tested	Test Inputs	Expected Outcome	Actual Outcome	Outcome
1a	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GB").	ui	human_bytes	1023	1023.0 B	1023.0 B	Pass
1b	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GB").	ui	human_bytes	-55	TypeError	-55.0 B	Fail
1c	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GB").	ui	human_bytes	0	0.0 B	0.0 B	Pass
1d	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GB").	ui	human_bytes	2000000000	1.9 GiB	1.9 GiB	Pass
1e	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GB").	ui	human_bytes	None	TypeError	TypeError	Pass

Fault Injection Results

Fault 1

Original Code	Code After Fault

def human_bytes(size):
"""Formats size, a number of bytes, in a human-readable way."""
powers = ['', 'K', 'M', 'G', 'T', 'P', 'E', 'Z', 'Y', 'H']
unit = 'B'
for power in powers:
if size < 1024:
return "%3.1f %s%s" % (size, power, unit)
size /= 1024.0
unit = 'iB'
return "big"
	def human_bytes(size):
	"""Formats size, a number of bytes, in a human-readable way."""
	powers = ['', 'K', 'M', 'G', 'T', 'P', 'E', 'Z', 'Y', 'H']
	unit = 'B'
	for power in powers:
	if size < 1024:
	return "%3.1f %s%s" % (size, power, unit)
	size /= 1024.0
	unit = 'iBytes'
	return "big"

1d	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GB").	ui	human_bytes	2000000000	1.9 GiB	1.9 GiB	Pass

1d	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GB").	ui	human_bytes	2000000000	1.9 GiB	1.9 GiBytes	Fail

Fault Injection Results

Fault 2

Original Code

```
def human_seconds(interval):
    """Formats interval, a number of seconds, as a human-readable time
    interval using English words.
    """
    units = [
        (1, 'second'),
        (60, 'minute'),
        (60, 'hour'),
        (24, 'day'),
        (7, 'week'),
        (52, 'year'),
        (10, 'decade'),
    ]
    for i in range(len(units) - 1):
        increment, suffix = units[i]
        next_increment, _ = units[i + 1]
        interval /= float(increment)
        if interval < next_increment:
            break
    else:
        # Last unit.
        increment, suffix = units[-1]
        interval /= float(increment)

    return "%3.1f %ss" % (interval, suffix)
```

Code After Fault

```
def human_seconds(interval):
    """Formats interval, a number of seconds, as a human-readable time
    interval using English words.
    """
    units = [
        (1, 'second'),
        (60, 'minute'),
        (60, 'hour'),
        (24, 'day'),
        (7, 'week'),
        (52, 'year'),
        (10, 'decade'),
    ]
    for i in range(len(units) - 1):
        increment, suffix = units[i]
        next_increment, _ = units[i + 1]
        interval /= float(increment)
        if interval > next_increment:
            break
    else:
        # Last unit.
        increment, suffix = units[-1]
        interval /= float(increment)

    return "%3.1f %ss" % (interval, suffix)
```

2a	The method human_seconds should take in an integer corresponding to a number of seconds and output a human readable string displaying the time interval using English words.	ui	human_seconds	600	10.0 minutes	10.0 minutes	Pass
2b	The method human_seconds should take in an integer corresponding to a number of seconds and output a human readable string displaying the time interval using English words.	ui	human_seconds	0	0.0 seconds	0.0 seconds	Pass
2c	The method human_seconds should take in an integer corresponding to a number of seconds and output a human readable string displaying the time interval using English words.	ui	human_seconds	583	9.7 minutes	9.7 minutes	Pass

2a	The method human_seconds should take in an integer corresponding to a number of seconds and output a human readable string displaying the time interval using English words.	ui	human_seconds	600	10.0 minutes	600.0 seconds	Fail
2b	The method human_seconds should take in an integer corresponding to a number of seconds and output a human readable string displaying the time interval using English words.	ui	human_seconds	0	0.0 seconds	0.0 decades	Fail
2c	The method human_seconds should take in an integer corresponding to a number of seconds and output a human readable string displaying the time interval using English words.	ui	human_seconds	583	9.7 minutes	583.0 seconds	Fail

Fault Injection Results

Fault 3

Original Code

```
def human_seconds_short(interval):
    """Formats a number of seconds as a short human-readable M:SS
    string.
    """
    interval = int(interval)
    return u'%i:%02i' % (interval // 60, interval % 60)
```

Code After Fault

```
def human_seconds_short(interval):
    """Formats a number of seconds as a short human-readable H:M.SS
    string.
    """
    interval = int(interval)
    hour = interval // 3600
    return u'%i:%i.%02i' % (hour, interval // 60, interval % 60)
```

3a	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	0	0:00	0:00	Pass
3b	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	153	2:33	2:33	Pass
3c	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	-55	TypeError	-1:05	Fail
3d	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	None	TypeError	Error	Fail
3e	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	5.5	0:05	0:05	Pass

3a	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	0	0:00	0:0.00	Fail
3b	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	153	2:33	0:2.33	Fail
3c	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	-55	TypeError	-1:-1.05	Fail
3d	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	None	TypeError	Error	Fail
3e	The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.	ui	human_seconds_short	5.5	0:05	0:0.05	Fail

An Automated Testing Framework: Designed for the Beets Open Source Software

Team Isn'tThisFun
CSCI 362-03

Zachary Davis, Kaitlyn Fulford, Ben Byrd, and Adam Sugarman

College of Charleston

Objective

Our goal for the project was to plan and implement an automatic testing framework for open source projects. We applied this framework to Beets, testing five different methods and injecting faults into the Beets source code to determine whether the framework catches bugs properly. The test results, pass or fail, were displayed on an HTML page.

Test Cases and Recording Procedures

Each test case specifies eight pieces of information necessary to run its test. The image below shows each piece. The readFiles() and parsefiles() methods of our script run through each of these test cases and process them for use. Test cases were recorded in a table with columns, test ID, method name, inputs, expected outcomes, actual outputs, and results (pass or fail). This table was displayed in an HTML file and saved.

```
#1. test number or ID
#2. requirement being tested
#3. component being tested
#4. method being tested
#5. test input(s) including command-line argument(s)
#6. expected outcome(s)
#7. driver name
#8. input type
```

Figure 1. The test case specification.

Results

The script builds an HTML table that displays test case info and test results. The test result (pass/fail) is based on the return of the compareTo() function in our script. We use browseLocal() to display the report in a browser window. The report is then saved to a 'temp' directory, where it can then be saved to 'reports'. The temp directory is cleared at the beginning of every testing run.

Test #	Req. Tested	Component Tested	Method Tested	Test Inputs	Expected Outcome	Actual Outcome	Outcome
1a	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GiB").	ui	human_bytes	1023	1023.0 B	1023.0 B	Pass
1b	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GiB").	ui	human_bytes	-55	TypeError	-55.0 B	Fail
1c	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GiB").	ui	human_bytes	0	0.0 B	0.0 B	Pass
1d	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GiB").	ui	human_bytes	2000000000	1.9 GiB	1.9 GiB	Pass
1e	The method human_bytes should format file size in a human readable way as a string (e.g. "1.9 GiB").	ui	human_bytes	None	TypeError	TypeError	Pass

Figure 2. The results of five test cases for a tested method.

Driver

We wrote a default driver which is used by all of our test cases. The method getAttr() calls the requisite function from the beets package, and returns the output to the script. Our driver has specialized logic to determine how many parameters a function takes, and adjusts the method signature appropriately for the call..

```
def driverDefaultFunc(info):
    """Calls the function specified in the test case specification file with the specified inputs,
    then returns the output."""

    inFuncName = info[3]
    inInputVal = info[4]

    ## If there is only one input value, call the function with one argument
    if (len(inInputVal) == 1):
        try:
            output = getattr(ui, inFuncName)(inInputVal[0])
        except TypeError as e:
            output = "TypeError"
        # except InputError:
        #     output = "InputError"
        except ValueError as e:
            output = "ValueError"
        except Exception, e:
            output = "Error"
    ## If there are two input values, call function with two arguments
    elif (len(inInputVal) == 2):
        try:
            output = getattr(ui, inFuncName)(inInputVal[0], inInputVal[1])
        except TypeError as e:
            output = "TypeError"
        except AssertionError as e:
            output = "AssertionError"
        except Exception, e:
            output = "Error"
    else:
        output = "Some Error"

    return output
```

Figure 3. The driver code.

Script

The Python script, runAllTests.py, is the backbone of our testing suite and is responsible for all of the logic that drives our project. runAllTests gathers information about each test case from test case files. This information is passed to the driver which runs the method and returns the output. The script then compares the actual output to the expected output. If they are equivalent, the test is passes. Otherwise, it is a fails. All of this information is then compiled into an HTML report and displayed in a browser window.

```
def main():
    readFiles()

def readFiles():
    """Reads in the lines from the test case specification file, passes them to
    parse, passes parsed list to driver specified by the test case specification
    file, and then calls the report function to handle building and displaying the
    results of the test in an HTML page."""

def parseFiles(inLineArray):
    """Parse the list of lines from the test case specification file to put these
    lines in the correct format."""

def compare(oracle, actualOutput):
    """Test whether the expected value from the test case specification file equals
    the actual output from the function."""

def strToFile(text, filename):
    """Write a file with the given name and the given text."""

def browseLocal(webpageText, filename='tempBrowseLocal.html'):
    """Start your webbrowser on a local file containing the text
    with given filename."""

def report(returnVal, contents2, outputVal):
    """ Write results of test to HTML file """
```

Figure 4. The signatures and doc-strings of the script file.

Conclusion

The team's testing framework analyzed five methods from an open source software called Beets. It displayed a pass or fail for each test case in an HTML report. The results of our testing determined that Beets has a robust code base. Most of the test cases passed, with the few fails occurring due to a lack of input sanitation..

Acknowledgments

Team Isn'tThisFun acknowledges Dr. Bowring for his aid in completing this project according to specification and professional standards and Ian Sommerville for the test plan structure.

Experiences and Lessons Learned

- Gained more Python experience
- Gained leadership skills
- Gained experience with version control and working with GitHub.
- Gained team work skills necessary along with communication skills
- Learned that a team has to be able to trust each other fully.