Ministry of Education of Republic of Moldova
Technical University of Moldova
CIM Faculty

Report

on PSI

Laboratory Work #3

Theme: Code analysis and improvement

Done by: Brailean Dumitru, st. of FAF-091 gr.
Verified by: Zarea Ivan, lector

Chişinău 2012

Table of Contents

1. Code inspection tools

1.1. Code Analyzer

1.1.1. PyCAna (Python Code Analyzer)

PyCAna (Python Code Analyzer) is a fancy name for a simple code analyzer for python that creates a class diagram after executing your code.

1.1.2. Pyntch

Pyntch (pron. "*pinch*", originally means PYthoN Type CHecker) is a static code analyzer for Python programming language. It detects possible runtime errors before actually running a code. Pyntch examines a source code and infers all possible types of variables, attributes, function arguments, and return values of each function or method. Then it detects possible exceptions caused by type mismatch, attribute not found, or other types of exceptions raised from each function. Pyntch does not address style issues. Pyntch normally runs pretty fast. It can perform checking for tens of thousands of lines of code within a minute.

1.2. Standard Compliance verification - PEP8 (Style Guide for Python Code)

1.2.1. Pylint

The Python community has formalized some recommended programming styles to help everyone write code in a common, agreed-upon style that makes the most sense for shared code. This style is captured in PEP-8. Pylint can be a quick and easy way of seeing if your code has captured the essence of PEP-8 and is therefore 'friendly' to other potential users.

1.3. Metrics

1.3.1. PyMetrics

PyMetrics produces metrics for Python programs. Metrics include McCabe's Cyclomatic Complexity metric (Cyclomatic complexity (CC) is the measure of linearly independent paths through a program), LoC, %Comments, etc. Users can also define their own metrics using data from PyMetrics. PyMetrics optionally outputs stdout, SQL command files and CSV

1.4. Code Coverage

1.4.1. Code Coverage

There are several subconcepts of Code coverage, which is just a quantitative measure of finding out how much of the code has been executed. The concept can be deceptive, though, if one doesn't know exactly what those figures mean.

For example, all of the following are subconcepts of code coverage: Statement coverage, Line coverage, Condition coverage, Decision coverage, Multiple condition coverage, Path coverage, etc.

A. Source Code for Inspection by PyLint:

```
import os
import sys
import datetime
import MySQLdb
import unittest

class AbstractTransactionManager(object):
	def __init__(self):
		raise NotImplementedError

	def StoreTransaction(self, transaction):
		raise NotImplementedError

	def GetTransactionById(self, transactionId):
		raise NotImplementedError

	def ToggleTransactionStatus(self, transactionId):
		raise NotImplementedError

	def CreateDb(self):
		raise NotImplementedError

"""
DROP TABLE IF EXISTS transactions CASCADE;

CREATE TABLE transactions (
	id BIGINT(99) NOT NULL AUTO_INCREMENT,
	originalId TEXT,
	phoneNumber TEXT,
	apId TEXT,
	originalRequest LONGTEXT,
	transformedRequest LONGTEXT,
	ttimestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
	sstatus INTEGER,
	UNIQUE KEY (id, ttimestamp) )
PARTITION BY RANGE (UNIX_TIMESTAMP(ttimestamp)) (
	PARTITION transactions_01_2012 VALUES LESS THAN (UNIX_TIMESTAMP('2012-01-01 00:00:00')) );

ALTER TABLE transactions ADD PARTITION ( PARTITION transactions_09_2012 VALUES LESS THAN ( UNIX_TIMESTAMP('2012-10-01 00:00:00') )
);

ALTER TABLE transactions ADD PARTITION ( PARTITION transactions_10_2012 VALUES LESS THAN ( UNIX_TIMESTAMP('2012-11-01 00:00:00') )
);
```

```
ALTER TABLE transactions ADD PARTITION ( PARTITION transactions_11_2012 VALUES LESS THAN ( UNIX_TIMESTAMP('2012-12-01 00:00:00') )
);

INSERT INTO transactions VALUES ( NULL, '123', '456', '789', '<foo_source>...</foo_source>',
'<foo_updated>...</foo_updated>',CURRENT_TIMESTAMP,0);

SELECT * FROM transactions;

SELECT * FROM transactions WHERE ttimestamp BETWEEN '2012-11-01 00:00:00' and '2012-12-01 00:00:00';

ALTER TABLE transactions DROP PARTITION transactions_11_2012;
"""

STATUS_PENDING = 0
STATUS_PROCESSED = 1

class MySQLTransactionManager(AbstractTransactionManager):
        def __init__(self):
                self.Connect()

        def __del__(self):
                self.Close()

        def Connect(self):
                try:
                        for i in range(3):
                                try:
                                        self.conn = MySQLdb.connect(host='localhost', user='root', passwd='123456', db='test')
                                        self.curs = self.conn.cursor()
                                        break
                                except MySQLdb.OperationalError, err:
                                        raise RuntimeError (err)
                except:
                        err = 'No servers to connect to!'
                        raise RuntimeError (err)

        def Close(self):
                print 'Closing MySQL connection'
                self.curs.close()
                self.conn.close()

        def CreateDB(self):
                try:
                        #self.curs.execute("""DROP TABLE IF EXISTS transactions CASCADE;""")
                        self.curs.execute("""
                                CREATE TABLE transactions (
                                        id                                                      BIGINT(99) NOT NULL
AUTO_INCREMENT,
                                        originalId                      TEXT,
                                        phoneNumber                             TEXT,
                                        apId                                    TEXT,
                                        originalRequest                 LONGTEXT,
                                        transformedRequest      LONGTEXT,
                                        ttimestamp                              TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,
                                        sstatus                                 INTEGER,
                                        UNIQUE KEY (id, ttimestamp) )
                                PARTITION BY RANGE (UNIX_TIMESTAMP(ttimestamp)) (
                                        PARTITION transactions_01_2000 VALUES LESS THAN (UNIX_TIMESTAMP('2000-01-01
00:00:00')) );
                                """)
                        self.conn.commit()

                except Exception, err:
                        raise RuntimeError (err)
```

```
	def PartitionDB(self, year = None, month = None):
		if not year:
			year = datetime.date.today().year
		if not month:
			month = datetime.date.today().month

		if month == 12:
			year_interval = year + 1
			month_interval = 1
		else:
			year_interval = year
			month_interval = month + 1

		try:
			self.curs.execute("""
				ALTER TABLE transactions ADD PARTITION (
					PARTITION transactions_%s_%s VALUES LESS THAN ( UNIX_TIMESTAMP('%s-%s-01
00:00:00') ) );
				""", (month, year, year_interval, month_interval))
			self.conn.commit()

		except Exception, err:
			raise RuntimeError (err)

	def StoreTransaction(self, t):
		for i in range(3):
			try:
				self.curs.execute("""
					INSERT INTO transactions
					VALUES (NULL,
					%s,%s,%s,%s,%s,CURRENT_TIMESTAMP,%s);""",
					(t['originalId'],
					t['phoneNumber'],
					t['apId'],
					t['originalRequest'],
					t['transformedRequest'],
					STATUS_PENDING,))
				self.conn.commit()
				return self.curs.lastrowid

			except Exception, err:
				code, msg = err
				if code == 1526:
					self.conn.rollback()
					self.PartitionDB()
				else:
					raise RuntimeError (err)

if __name__ == '__main__':
	mysql = MySQLTransactionManager()
	#mysql.CreateDB()
	t = {'originalId': 12345,
		'phoneNumber': 12345,
		'apId': 2222,
		'originalRequest': '<foo_source>...</foo_source>',
		'transformedRequest': '<foo_updated>...</foo_updated>'}
	print mysql.StoreTransaction(t)
```

B. Inspection result

Message	File Name	Line	Position
[C] Invalid name "MYSQLT~1" (should match (([a-z_][a-z0-9_]*)|([A-Z][a-zA-Z0-9]+))$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	1	
[C] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	1	
[C, AbstractTransactionManager] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	7	
[C, AbstractTransactionManager.StoreTransaction] Invalid name "StoreTransaction" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	11	
[C, AbstractTransactionManager.StoreTransaction] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	11	
[C, AbstractTransactionManager.GetTransactionById] Invalid name "GetTransactionById" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	14	
[C, AbstractTransactionManager.GetTransactionById] Invalid name "transactionId" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	14	
[C, AbstractTransactionManager.GetTransactionById] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	14	
[C, AbstractTransactionManager.GetTransactionById] Invalid name "transactionId" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	14	
[C, AbstractTransactionManager.ToggleTransactionStatus] Invalid name "ToggleTransactionStatus" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	17	
[C, AbstractTransactionManager.ToggleTransactionStatus] Invalid name "transactionId" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	17	
[C, AbstractTransactionManager.ToggleTransactionStatus] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	17	
[C, AbstractTransactionManager.ToggleTransactionStatus] Invalid name "transactionId" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	17	
[C, AbstractTransactionManager.CreateDb] Invalid name "CreateDb" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	20	
[C, AbstractTransactionManager.CreateDb] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	20	
[W] String statement has no effect	D:\FAF\SEM VII\PSI\labs\lab#3\m...	52	
[W, MySQLTransactionManager] Method 'CreateDb' is abstract in class 'AbstractTransactionManager' but is not overridden	D:\FAF\SEM VII\PSI\labs\lab#3\m...	57	
[W, MySQLTransactionManager] Method 'GetTransactionById' is abstract in class 'AbstractTransactionManager' but is not overridden	D:\FAF\SEM VII\PSI\labs\lab#3\m...	57	
[W, MySQLTransactionManager] Method 'ToggleTransactionStatus' is abstract in class 'AbstractTransactionManager' but is not overridden	D:\FAF\SEM VII\PSI\labs\lab#3\m...	57	
[C, MySQLTransactionManager] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	57	
[W, MySQLTransactionManager.__init__] __init__ method from base class 'AbstractTransactionManager' is not called	D:\FAF\SEM VII\PSI\labs\lab#3\m...	58	
[C, MySQLTransactionManager.Connect] Invalid name "Connect" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	64	
[C, MySQLTransactionManager.Connect] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	64	
[W, MySQLTransactionManager.Connect] Unused variable 'i'	D:\FAF\SEM VII\PSI\labs\lab#3\m...	66	
[C, MySQLTransactionManager.Close] Invalid name "Close" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	77	
[C, MySQLTransactionManager.Close] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	77	
[C, MySQLTransactionManager.CreateDB] Invalid name "CreateDB" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	82	
[C, MySQLTransactionManager.CreateDB] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	82	
[C, MySQLTransactionManager.PartitionDB] Invalid name "PartitionDB" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	104	
[C, MySQLTransactionManager.PartitionDB] Missing docstring	D:\FAF\SEM VII\PSI\labs\lab#3\m...	104	
[W, MySQLTransactionManager.StoreTransaction] Redefining name 't' from outer scope (line 154)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	127	
[C, MySQLTransactionManager.StoreTransaction] Invalid name "StoreTransaction" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	127	
[C, MySQLTransactionManager.StoreTransaction] Invalid name "t" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	127	
[C, MySQLTransactionManager.StoreTransaction] Invalid name "t" (should match [a-z_][a-z0-9_]{2,30}$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	127	
[W, MySQLTransactionManager.StoreTransaction] Catching too general exception Exception	D:\FAF\SEM VII\PSI\labs\lab#3\m...	143	
[W, MySQLTransactionManager.StoreTransaction] Unused variable 'msg'	D:\FAF\SEM VII\PSI\labs\lab#3\m...	144	
[W, MySQLTransactionManager.StoreTransaction] Unused variable 'i'	D:\FAF\SEM VII\PSI\labs\lab#3\m...	128	
[W, MySQLTransactionManager.Connect] Attribute 'curs' defined outside __init__	D:\FAF\SEM VII\PSI\labs\lab#3\m...	69	
[W, MySQLTransactionManager.Connect] Attribute 'conn' defined outside __init__	D:\FAF\SEM VII\PSI\labs\lab#3\m...	68	
[C] Invalid name "mysql" (should match (([A-Z_][A-Z0-9_]*)|(__.*__))$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	152	
[C] Invalid name "t" (should match (([A-Z_][A-Z0-9_]*)|(__.*__))$)	D:\FAF\SEM VII\PSI\labs\lab#3\m...	154	
[W] Unused import os	D:\FAF\SEM VII\PSI\labs\lab#3\m...	1	
[W] Unused import unittest	D:\FAF\SEM VII\PSI\labs\lab#3\m...	5	
[W] Unused import sys	D:\FAF\SEM VII\PSI\labs\lab#3\m...	2	
[R, AbstractTransactionManager] Abstract class is only referenced 1 times	D:\FAF\SEM VII\PSI\labs\lab#3\m...	7	

2. Refactoring

2.1. Add Parameter Pattern

Add a parameter for an object that can pass on this information.

Example:

```
def StoreTransaction(self, t):
```

This pattern is essential to my code due to the fact that StoreTransaction method provides the operational logic that must be applied on a set of data that varies constantly, there for that data is passed to the function via a parameter.

2.2. Consolidate Duplicate Conditional Fragments Pattern

The same fragment of code is in all branches of a conditional expression. Move it outside of the expression.

Example:

```
if month == 12:
        year_interval = year + 1
        month_interval = 1
else:
        year_interval = year
        month_interval = month + 1
```

The reason for using this pattern is defined by the fact that more chec condition the code has more time it takes to execute it.

2.3. Decompose Conditional Pattern

Extract methods from the condition, then part, and else parts.

Example:

```
if not year:
        year = datetime.date.today().year
if not month:
        month = datetime.date.today().month

if month == 12:
        year_interval = year + 1
        month_interval = 1
else:
        year_interval = year
        month_interval = month + 1
```

Reason for this pattern to use is the fact that the more complicated the check condition are the more probable is the fact that an error may acure in the code, and the testing is harder to do and more complex.

2.4. Extract Interface Pattern

Example:

```
class AbstractTransactionManager(object):
```

The use of this pattern offer the modularity for the project and in the future development of the project will offer the pofibility to add other implementations like PostgreSQL.

2.5. Replace Error Code with Exception

Example:

```
except Exception, err:
        code, msg = err
        if code == 1526:
                self.conn.rollback()
                self.PartitionDB()
        else:
                raise RuntimeError (err)
```

In here the exception code that are obtained from the interaction with the data base is treated if possible and if not thy are raised to the next level so that an error that accrues at this level of system will stop the execution of other and omit a bigger error from accruing in the system.

3. Architectural patterns

3.1. Service Layer

Defines an application's boundary with a layer of services that establishes a set of available operations and coordinates the application's response in each operation.

This principle is a core one used in the bigger picture project, and that is observed by the fact that at this level I implemented the systems deals only with database and sends the result to a higher level of application.

3.2. Data Mapper

A layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself.

The module I have implemented is the Data Mapper that deals derectly with the database processes queries and sends data to the upper level modules in the required form.

3.3. Front Controller

A controller that handles all requests for a Web site.

This part is implemented by a another module that interacts directly with the web interface and passes request to my module to be processed and stored in to the database.