**Testowanie aplikacji w Ruby**

(pracownia komputerowa – 45 godz.)

1. (3 godz.) Przygotowanie środowiska do pracy z językiem Ruby.

* Instalacja Git, RVM.
* Integracja z usługami na GitHub – Travis, Code Climate, Coveralls.

2. (6 godz.) Tworzenie prostych skryptów w języku Ruby.

3. (6 godz.) Dopisywanie brakującego kodu do nieprzechodzących testów.

4. (6 godz.) Dopisywanie testów do niepokrytego nimi kodu.

5. (9 godz.) *Agile Programming*.

* Test first / Test Last
* TDD (*Test Driven Development*) w przykładach.

6. (12 godz.) Własny projekt do napisania z wykorzystaniem TDD lub dowolny projekt z GitHub do którego dopisano testy (i kod) zaakceptowane przez właściciela projektu.

7. (3 godz.) Prezentacje najlepszych projektów.

**Materiały do pracowni komputerowej**

materiały można pobrać z repozytorium:

*https://github.com/projekty/syllabus-testowanie-aplikacji/tree/master/pracownie*

2. Tworzenie prostych skryptów w języku Ruby

W katalogu 02/pdf i 02/github znajduje się kilka skryptów.

Zadanie 0. (Czytanie kodu ze zrozumieniem.) Co robią te skrypty?

Zadanie 1. *Bundlerize scripts,* oznacza to że należy dodać plik Gemfile

w którym wpiszemy użyte gemy w najnowszych wersjach.

Następnie uaktualnimy kod do tych wersji, tak aby skrypty działały.

Zadaniee 2. Polega na dopisaniu testów. Zaczynamy od skryptu

02/pdf/lista.rb.

require "prawn"

require "csv"

require "pp"

Prawn::Document.generate("lista.pdf") do

# font "#{Prawn::BASEDIR}/data/fonts/DejaVuSans.ttf"

font "/usr/share/fonts/dejavu/DejaVuSans.ttf"

font\_size 11

if ARGV.length == 0

puts "Usage: ruby lista.rb NAME.CSV"

exit

else

# csv\_file = "asi2011.csv"

csv\_file = ARGV[0]

end

# font "#{Prawn::BASEDIR}/data/fonts/DejaVuSans.ttf"

csv\_data = open(csv\_file).readlines

body = CSV.parse(csv\_data.join)

nazwisko = make\_cell :content => "nazwisko"

imie = make\_cell :content => "imię"

repo\_url = make\_cell :content => "url repo na github.com"

index = 0

counted = body.map do |row|

index += 1

row.unshift (index < 10 ? " " : "") + index.to\_s

row += [" ", " "]

end

counted.unshift ["", nazwisko, imie, repo\_url, "uwagi"]

table(counted, :header => true,

:width => 540,

:column\_widths => {0 => 32},

:cell\_style => {:height => 36, :padding => [12, 6, 12, 6], :overflow => :shrink\_to\_fit}

)

end

Następny skrypt do poprawek, to 02/github/last-commit:

begin

require 'ansi/code' # http://rubydoc.info/gems/ansi/frames

include ANSI::Code

require 'bitbucket\_rest\_api' # http://rubydoc.info/gems/bitbucket\_rest\_api/frames

require 'github\_api' # http://rubydoc.info/gems/github\_api/frames

require 'csv'

require 'time'

require 'optparse'

require 'ostruct'

require 'yaml'

# na podstawie http://github.com/wbzyl/nosql-tutorial/blob/master/pp/db/mongodb/gutenberg2mongo.rb

class ParseArgs

def self.parse(args)

options = OpenStruct.new

options.quiet\_mode = false # wyświetlać informacje w przypadku błędu 404 (nieodnaleziono użytkownika bądź repo)

options.group = false # filtrowanie po grupie

options.csv\_file = false # plik z danymi

options.message = false # wyświetlać wiadomość przesłaną razem z commitem?

options.user\_regex = false # filtrowanie po nazwie użytkownika

options.commits = 1 # ilość ostatnich commitów do wyświetlenia

options.error\_details = false # wyświetlać szczegółowy komunikat błędu?

options.credentials\_github = false # basic\_auth dla githuba

options.credentials\_bitbucket = false # basic\_auth dla bitbucketa

opts = OptionParser.new do |opts|

opts.banner = "Użycie: #{$0} plik.csv [OPCJE]"

opts.on("-q", "--quiet") do

options.quiet\_mode = true

end

opts.on("-g", "--group GRUPA", "grupa (pon12,wto10,wto12)") do |name|

options.group = name

end

opts.on("-m", "--message", "wyświetl wiadomość wysłaną z commitem") do

options.message = true

end

opts.on("-u", "--user REGEX", "wyrażenie regularne do filtrowania loginu") do |regex|

options.user\_regex = regex

end

opts.on("-c", "--commits NUMBER", "ilość comitów do wyświetlenia. Domyślnie 1, 0 aby wyświetlić maksymnalną ilość") do |number|

options.commits = number

end

opts.on("--error-details", "wyświetla szczegóły błędu") do

options.error\_details = true

end

opts.on("--credentials PATH", "folder w katalogu #{ENV['HOME']} zawierający plik github.yml oraz bitbucket.yml z polami 'login' i 'password'") do |path|

begin

options.credentials\_github = YAML.load\_file File.join(ENV['HOME'], path, '/github.yml')

rescue

puts red("Brak pliku " + yellow("#{ENV['HOME']}/#{path}/github.yml"))

end

begin

options.credentials\_bitbucket = YAML.load\_file File.join(ENV['HOME'], path, '/bitbucket.yml')

rescue

puts red("Brak pliku " + yellow("#{ENV['HOME']}/#{path}/bitbucket.yml"))

end

end

opts.separator "Pozostałe opcje:"

opts.on\_tail("-h", "--help", "wypisz pomoc") do

puts opts

exit

end

end

options.csv\_file = opts.parse!(args)[0]

options

end

end

options = ParseArgs.parse(ARGV)

if not options.csv\_file

puts "Pomoc: #{$0} --help"

exit

end

data = open(options.csv\_file).readlines

csv\_data = CSV.parse(data.join)

if options.credentials\_github && (options.credentials\_github['login'] && options.credentials\_github['password'])

github = Github.new login: options.credentials\_github['login'], password: options.credentials\_github['password']

else

github = Github.new

end

if options.credentials\_bitbucket && (options.credentials\_bitbucket['login'] && options.credentials\_bitbucket['password'])

bitbucket = BitBucket.new login: options.credentials\_bitbucket['login'], password: options.credentials\_bitbucket['password']

else

bitbucket = BitBucket.new

end

csv\_data.each do |repo|

if options.user\_regex && !(repo[2].downcase =~ /#{options.user\_regex}/)

next

end

begin

array = Array.new

if repo[4]=="bitbucket"

all = bitbucket.repos.changesets.all(repo[2], repo[3])

for i in 0..(all.changesets.size-1)

row = all.changesets[i]

array.push(:date => row[:timestamp], :author => row[:raw\_author], :message => row[:message])

end

array.reverse

elsif repo[4]=="github"

github.repos.commits.list(repo[2], repo[3]).each do |row|

array.push(:date => DateTime.parse(row.commit.author[:date]).strftime("%F %T"), :author => row.commit.author[:name] + " <#{row.commit.author[:email]}>", :message => row.commit[:message])

end

else

puts "Zdefiniuj typ repozytorium dla konta #{repo[2]}"

next

end

rescue Github::Error::NotFound, BitBucket::Error::NotFound

if !options.quiet\_mode

puts "Nie znaleziono konta #{yellow(repo[2])} lub repozytorium #{yellow(repo[3])} dla #{red(repo[0])} #{red(repo[1])}"

end

next

end

if (options.commits.to\_i == 0) || (options.commit.to\_i >= array.size)

max = array.size-1

else

max = options.commits.to\_i-1

end

for i in 0..max

date = array[i][:date] < Date.today.prev\_day(8).to\_s ? red(array[i][:date]) : green(array[i][:date])

line = "#{date} #{yellow(repo[2] + '/' + repo[3])} (#{array[i][:author]})"

if options.message

line += " {" + cyan(array[i][:message]) + "}"

end

if options.group && options.group == repo[5]

puts line

elsif !options.group

puts line += " " + magenta(repo[5])

end

end

end

rescue SystemExit, Interrupt

puts "Operacja anulowana!"

rescue Github::Error::Forbidden, BitBucket::Error::Forbidden => e

puts red("Błąd przy próbie pobrania danych! prawdopodobnie wyczerpałeś limit zapytań")

puts "Szczegóły błędu: #{e}" if options.error\_details

rescue Exception => e

puts "Niezdefinowany błąd aplikacji!"

puts "Szczegóły błędu: #{e}" if options.error\_details

end

3. Dopisywanie brakującego kodu do nieprzechodzących testów.

Należy dopisać kod do testów 03/spec/part1\_spec.rb i 03/spec/part1\_spec.rb.

Plik part1\_spec.rb

require\_relative '../lib/part1'

describe "#sum" do

it "should be defined" do

expect { sum([1,3,4]) }.not\_to raise\_error

end

it "returns the correct sum" do

expect(sum([1,2,3,4,5])).to be\_a\_kind\_of Fixnum

expect(sum([1,2,3,4,5])).to eq(15)

expect(sum([1,2,3,4,-5])).to eq(5)

end

end

describe "#max\_2\_sum" do

it "should be defined" do

expect { max\_2\_sum([1,2,3]) }.not\_to raise\_error

end

it "returns the correct sum" do

expect(max\_2\_sum([1,2,3,4,5])).to be\_a\_kind\_of(Fixnum)

expect(max\_2\_sum([1,2,3,4,100])).to eq(104)

expect(max\_2\_sum([1,-2,-3,-4,-5])).to eq(-1)

end

end

describe "#sum\_to\_n?" do

it "should be defined" do

expect { sum\_to\_n?([1,2,3],4) }.not\_to raise\_error

end

it "returns the correct value" do

expect(sum\_to\_n?([1,2,3,4,5], 5)).to be true

expect(sum\_to\_n?([1,2,5,6,7,8], 3)).to be true

expect(sum\_to\_n?([100,50,50,2,100,4,5], 100)).to be true

expect(sum\_to\_n?([1,2,3,4,5], -3)).to be false

end

end

Plik part2\_spec.rb

require\_relative '../lib/part2'

# https://www.relishapp.com/rspec/rspec-expectations/v/3-0/docs/built-in-matchers

describe "#hello" do

it "should be defined" do

expect { hello("Testing") }.not\_to raise\_error

end

it "The hello method prints the correct string" do

expect(hello("Dan").class).to eq(String)

expect(hello("Dan")).to eq('Hello, Dan'),

"Incorrect results for input: \"Dan\""

end

end

describe "#starts\_with\_vowel?" do

it "should be defined" do

expect { starts\_with\_vowel?("b") }.not\_to raise\_error

end

it "The starts\_with\_vowel? method returns the correct boolean" do

expect(starts\_with\_vowel?("asdfgh")).to be(true),

"Incorrect results for input: \"asdfgh\""

expect(starts\_with\_vowel?("Veeeeeeee")).to be(false),

"Incorrect results for input: \"Veeeeeeee\""

end

end

describe "#binary\_multiple\_of\_4?" do

it "should be defined" do

expect { binary\_multiple\_of\_4?("yes") }.not\_to raise\_error

end

it "The binary\_multiple\_of\_4? method returns the correct boolean" do

expect(binary\_multiple\_of\_4?("111111101")).to be(false),

"Incorrect results for input: \"111111101\""

expect(binary\_multiple\_of\_4?("1010101010100")).to be(true),

"Incorrect results for input: \"1010101010100\""

end

end

4. Dopisywanie testów do niepokrytego nimi kodu.

Zadanie 1. Napisać metodę, która posortuje elementy według liczby

powtórzeń, zaczynając od wartości powtarzających się najrzadziej.

Metodę nazwać `sort\_by\_freq` i dodać ją do modułu Enumerable:

module Enumerable

def sort\_by\_freq

# ... dopisać brakujący kod

end

end

puts [1,2,3,4,1,2,4,8,1,4,9,16].sort\_by\_freq.join(", ")

# 3, 8, 9, 16, 2, 2, 1, 1, 1, 4, 4, 4

Zacząć od napisania testów w pliku *spec/part1\_spec.rb*.

2. Dla dużych tablic zwierających stosunkowo niewiele różnych

obiektów wyliczyć histogram obrazujący częstotliwość

występowań danego obiektu. Metodę nazwać `to\_hist`

i dodać ją do modułu Enumerable:

module Enumerable

def to\_hist

# ... dopisać brakujący kod

end

end

# [1, 3, 3, 4, 4, 4].to\_hist #=> {1=>1, 3=>2, 4=>3}

Zacząć od napisania testów w pliku *spec/part2\_spec.rb*.