API Testing Recipes in Ruby

The Problem Solving Guide to API Testing

Zhimin Zhan

API Testing Recipes in Ruby

The problem solving guide to API Testing

Zhimin Zhan

This book is for sale at http://leanpub.com/api-testing-recipes-in-ruby

This version was published on 2016-09-04

ISBN 978-1537344782

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By Zhimin Zhan

Practical Web Test Automation

Watir Recipes

Selenium WebDriver Recipes in Ruby

Selenium WebDriver Recipes in Java

Learn Ruby Programming by Examples

Learn Swift Programming by Examples

Selenium WebDriver Recipes in Python

Selenium WebDriver Recipes in Node.js

Contents

1. Introduction

API (Application Programming Interface) is a collection of software operations that can be executed by other software applications, instead of human users. For example, an online store takes payment from a customer's credit card. The payment processing is typically done via API calls to a payment gateway such as PayPal. In this case, the payment gateway provides API, and the online store application is a consumer of API.

API testing is to verify the functionality of APIs.

API Testing vs Unit Testing

Unit testing is a type of white box testing performed by programmers at source code level (I prefer the term *Programmer Test* than *Unit Test*, as it is created and maintained by programmers). Unit Testing falls under white-box testing, API testing is largely under black-box testing.

> ### White-box testing vs Black-box testing
>
> In white-box testing, the tester is much more concerned with internal operations of the application. In black-box testing, the tester is only concerned with the functionality of the overall application, exercising only the inputs and checking the outputs.

API Testing vs GUI Testing

The main difference between GUI testing and API testing is obvious: we don't see the application's user interface in pure API testing. Besides, API testing differs UI Testing in:

- **Fast**

  API testing is usually an order of magnitude faster than a similar functional UI test. For example, to submit user registration form on a web page, a functional UI test case needs to open a browser, fill every field and click the 'Submit' button. If it is done by

a web service API, just one single HTTP request with registration details typically in a XML or JSON.

- **More programming skills required**

  While good programming/scripting skills is required by API and functional UI testing (don't trust scriptless testing), people with no or little coding experience might still be able to get started with UI testing with the help of recorders. The nature of API testing, such as preparing test request data and parsing response data, requires more technical skills.

- **Less prone to changes**

  Comparing to GUI, changes to API is much less frequently. Even when an API-breaking change are unavoidable, providers will keep the current version (commonly renamed to `/api-v1/...`). However, developers usually won't have this kind of consideration when changing web pages.

Automated API Testing

Like functional testing, API testing can be done manually, and we see it is commonly done so. For example, to test a SOAP web service call, a test might perform the testing as below:

1. Open SoapUI tool
2. Create a new SOAP project and enter the initial WSDL (Web Services Description Language)
3. Select one service call and create a sample request
4. Edit the request XML
5. Invoke the web service
6. Verify the response returned

This is what I call manual API testing. Automated API testing is to invoke APIs by using automated test scripts. While we might use GUI tools to develop or debug test scripts, execution of test scripts (including verification) must be automated, no human intervention is required.

API testing substituted end-to-end functional testing

Ideally, full end-to-end functional testing via GUI is the best for ensuring software quality, if can be done successfully (*all functional UI tests executed many times a day with quick feedback*). However, few projects achieve this. Here is a text except from API Testing Wikipedia page[1]: "API testing is now considered critical for automating testing because APIs now serve as the primary interface to application logic and because GUI tests are difficult to maintain with the short release cycles and frequent changes commonly used with Agile software development and DevOps)." - Forrester 2015[2]

If you are interested in implementing end-to-end functional testing for web applications, read my other book Practical Web Test Automation[3].

An API test example

The test below invokes a web service to register a new user.

```
new_user_registration_xml  = <<END_OF_MESSAGE
<UserRegistration>
  <UserName>wisetester</UserName>
  <Email>testwisely01@gmail.com</Email>
  <Password>secret</Password>
</UserRegistration>
END_OF_MESSAGE

require 'httpclient'
http = HTTPClient.new
ws_url = "https://agileway.net/api/register"
resp = http.put(ws_url, new_user_registration_xml)
expect(resp.body).to include("Registration successful")
```

The above test posts a block of XML to register a new user. If it is done via UI, we need to follow many steps as this Selenium WebDriver test below:

[1] https://en.wikipedia.org/wiki/API_testing

[2] http://blogs.forrester.com/diego_lo_giudice/15-04-23-the_forrester_wave_evaluation_of_functional_test_automation_fta_is_out_and_its_all_about_going_be?cm_mmc=RSS-_-BT-_-63-_-blog_1769

[3] https://leanpub.com/practical-web-test-automation

```
driver = Selenium::WebDriver.for(:chrome)
driver.navigate.to("https://agileway.net")
driver.find_element(:link_text, "CREATE ACCOUNT").click
driver.find_element(:name, "email").send_keys("testwisely01@gmail.com")
driver.find_element(:name, "username").send_keys("wisetester")
driver.find_element(:name, "password").send_keys("secret")
driver.find_element(:name, "passwordConfirm").send_keys("secret")
driver.find_element(:xpath, "//input[@id='terms']/../i").click
driver.find_element(:id, "sign_up_btn").click
expect(driver.page_source).to include("Registration successful")
```

Skills required

Comparing to functional UI testing, writing API testing requires more technical skills. Without UI, we communicate with scripting.

- Knowledge of API and protocols

  Testers are required not only to know the top level protocol, but also the technologies or protocols underneath. For example, to write test scripts for SOAP web services, understanding HTTP, URL, XML and XPath is a must.
- Coding Skills

  Flexible API test scripts are in a syntax of programming language, such as Ruby and Python. To effectively develop and maintain test scripts, mastering programming concept and the language is a must. Having said that, the level required, at least at the beginning, for API testing is not as high as programmers.
- Parsing data data structures

  The messages (request and response) in API testing are commonly in XML and JSON format. Obviously, a good knowledge of them is a must. For example, parsing XML document using an XML processor and extract a specific element using XPath.

Use Ruby

API testing requires programming knowledge. Ruby is a dynamic, open source scripting language with a focus on simplicity and productivity, which make it ideal for writing test scripts. The below are three books on testing with Ruby from one publisher:

- "Scripted GUI Testing with Ruby"[4]
- "Everyday Scripting with Ruby: For Teams, Testers, and You"[5]
- "Continuous Testing with Ruby, Rails, and JavaScript"[6]

You might have heard of some popular testing frameworks in Ruby, such as:

- Watir - Web Application Testing in Ruby[7]
- Cucumber[8]
- Capybara[9]

The use of Ruby in test scripts does not mean we can only test services written in Ruby, not at all. In last 5 years, I have written API automated tests (all in Ruby) for APIs coded in Java and C#. The word API means an agreed standard. Take RESTful web services as an example, requests are sent to server by URL, responses are in XML or JSON format. It does not matter which language was used to implement the API.

Ruby - Most valuable programming language

Based on this Quartz report[10], Ruby is also "the most valuable programming skills to have on a resume".

Install Ruby

- Windows: Ruby Installer for Windows[11]
- Mac OS X: Ruby 2.0 is included.
- Linux: can be easily installed, typically runs one command using a package manager

Add to your PATH.

[4] https://pragprog.com/book/idgtr/scripted-gui-testing-with-ruby
[5] https://pragprog.com/book/bmsft/everyday-scripting-with-ruby
[6] https://pragprog.com/book/rcctr/continuous-testing
[7] http://watir.com/
[8] https://cucumber.io/
[9] http://jnicklas.github.io/capybara/
[10] http://qz.com/298635/these-programming-languages-will-earn-you-the-most-money/
[11] http://rubyinstaller.org/

Install Gem

Ruby gems (a cool name for libraries in Ruby) are centrally hosted at rubygems.org[12]. To install or update a ruby gem, you need to be connected to Internet.

```
> gem install mail
```

The above command (run from a command line window) will download and install latest mail gem. The command below lists all the gems installed on your machine.

```
> gem list
```

RSpec Test Framework

To make the effective use of scripts for testing, we need to put them in a test framework that defines test structures and provides assertions (performing checks in test scripts). Typical choices are:

- xUnit Unit Test Frameworks such as JUnit (for Java), NUnit (for C#) and minitest (for Ruby).
- Behaviour Driven Frameworks such as RSpec and Cucumber (for Ruby).

In this book, I use RSpec, the de facto Behaviour Driven Development (BDD) framework for Ruby. Here is an example.

```
describe "REST WebService" do

  it "REST - List all records" do
    http = HTTPClient.new
    resp = http.get("http://www.thomas-bayer.com/sqlrest/CUSTOMER")
    xml_doc = REXML::Document.new(resp.body)
    expect(xml_doc.root.elements.size).to be > 10
  end
```

[12] https://rubygems.org

```
  it "REST - Get a record" do
    http = HTTPClient.new
    resp = http.get("http://www.thomas-bayer.com/sqlrest/CUSTOMER/4")
    expect(resp.body).to include("<CITY>Dallas</CITY>")
  end

end
```

The keywords `describe` and `it` define the structure of a test script.

- **`describe "..." do`**

  Description of a collection of related test cases
- **`it "..." do`**

  Individual test case.

`expect().to` statements are called rspec-expectations, which are used to perform checks (also known as assertions).

You will find more about RSpec from its home page[13]. However, I honestly don't think it is necessary. The part used for test scripts is not much and quite intuitive. After studying and trying out some examples, you will be quite comfortable with RSpec.

Execution of RSpec tests is covered in Chapter 11.

Avoid SoapUI alike tools

This might send a shock for some readers, for them, SoapUI might be equivalent to API testing. In my opinion, working on test scripts in plain text directly is more flexible and simpler, which are important factors for test maintenance. Some will say, SoapUI does have scripting capability as well, using Groovy, a dynamic scripting language for the Java platform. Yes, that is true, but not good enough.

As we know, the real challenge of test automation (functional UI and API Testing) is not about writing the test, it is the maintenance of **all** regression tests. Please note the word "all", I highlight it here to show a perspective of API testing. If the maintenance of test scripts is hard, hiding the test scripts with a GUI layer is not a good approach. Instead, work on the test scripts directly to make them flexible by using programming techniques. Most software are written in this way, we certainly can do with the test scripts.

[13] http://rspec.info

The Win of plain text scripting for Selenium WebDriver

Nowadays, Selenium WebDriver undisputedly dominates functional testing for web applications. It was not the case 6 years ago. At that time, I found myself quite lonely at software testing conferences promoting open-source test scripting frameworks such as Watir and Selenium, surrounded by talks on fancy and expensive scriptless testing tools. Now many commercial functional testing tool vendors (including SmartBear, the new owner of SoapUI) have changed to support Selenium WebDriver. Bear in mind that all these commercial tools have some of own proprietary scripting as well under their GUI tool. Those script syntax were all lost to Selenium WebDriver for functional testing web apps. The main reasons of the success of Selenium WebDriver, in my opinion, are:

- text based
- empowered by programming
- good support by browser vendors

Comparing to Selenium WebDriver UI testing, developing test scripts for API testing is more like programming.

Developing plain text scripts does not mean typing them in NotePad. Like programmers, we do it in Integrated Development Environment (IDE, such as TestWise) or code editors (such as Sublime Text). This does not mean excluding helps from other GUI tools either. In fact, in next chapter, I will show you how to use free SoapUI open source edition to generate SOAP request templates.

The benefits of text scripting

The classic "The Pragmatic Programmer, From Journeyman to Master" book dedicated one chapter on "The Power of Text". If you haven't read it, I strongly recommend you to do so.

- **Integrate with other tests**

  For example, we can test a SOAP web service call, then go to the target web site to verify the content on a web page using Selenium WebDriver, all in one test case.
- **Easy to run**

  By using a build tool of a chosen scripting language, we can easily customize test executions, such as a suite of selected tests or test files in a pattern.

- **Flexible**

  Ruby is a powerful script language, and there are many libraries (called Gems) we can use. For example, for test scripts in RSpec, we can set certain test steps to run before and after each test case. For assertion, if we want to verify an element's value returned in SOAP is today's date, it is going to be quite difficult in SoapUI. In Ruby, it can be as simple as `expect(elem.text).to eq(today)`.
- **Script reuse**

  Ruby is a full-featured object oriented programming language. By using inheritance and modules, we can maximum the script reuse for much more maintainable test scripts.
- **Integrate with Continuous Integration**

  If we use a standard scripting (and open) language, it will be easier to integrate with CI servers. We can trigger an execution of all API tests with a click of button and get test report including historical results of each individual test. We can even distribute tests to multiple build machines to run them in parallel to greatly reduce the execution time.

Testing Tool

For API testing, the most involved tool is IDE or coding editor.

- Syntax highlight and validation
- Efficient Ruby code editing
- Flexible test execution and debugging

The tools Ruby programmers and testers use generally fall in the two categories as below:

Integrated Development Environment (IDE)

- Apatana[14], free
- NetBeans 6.9[15], free
- RubyMine[16], commercial

[14] http://www.aptana.com
[15] https://netbeans.org/downloads/6.9.1
[16] https://www.jetbrains.com/ruby/

Code Editor

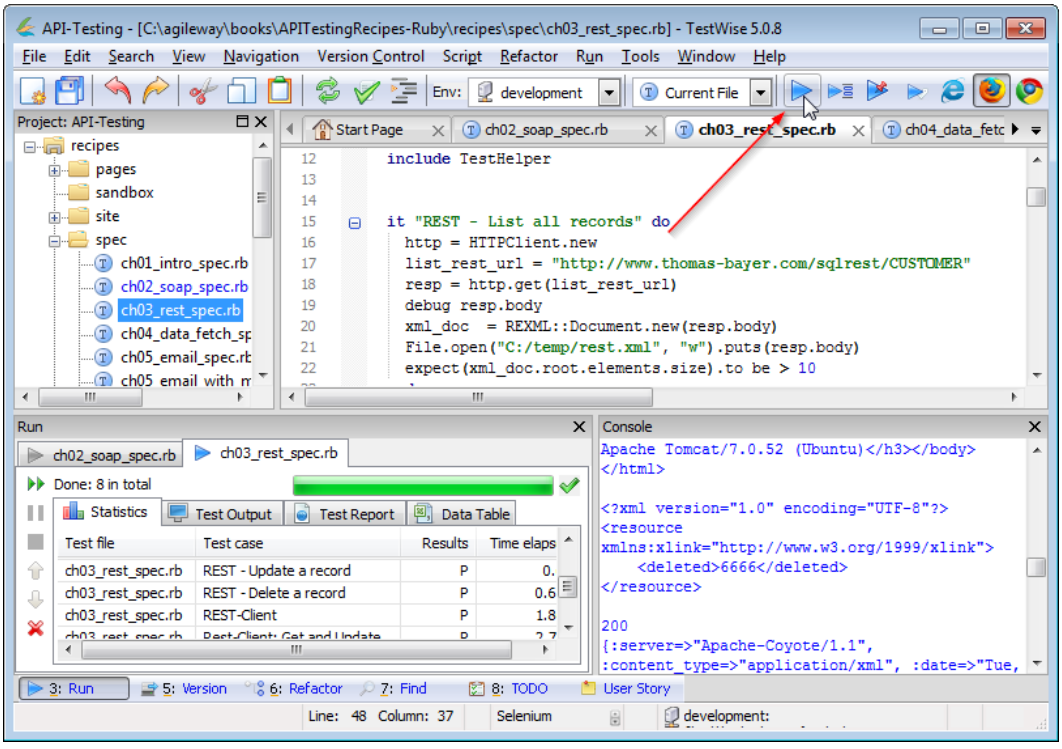
- Atom[17], free
- TextMate[18], commercial, Mac only
- Sublime Text[19], commercial
- Visual Studio Code[20], free
- Vim[21], free

As you can see, there are many choices here. I use TestWise 5[22] (disclaimer: I created TestWise. The Pro edition is a commercial software). TestWise is a testing IDE that supports Selenium-WebDriver for testing web applications. I usually develop API tests and UI tests in one project, quite often, they benefit each other.

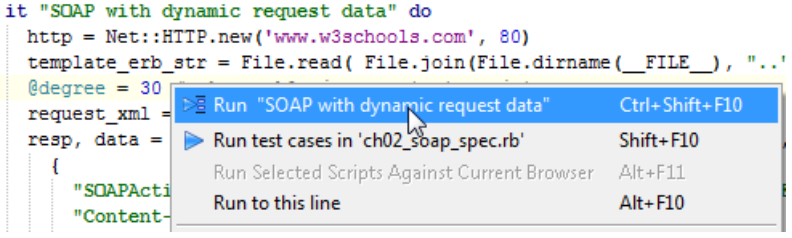
Run test in TestWise

To run all test cases in a test script file.

[17] https://atom.io/
[18] https://macromates.com/
[19] http://www.sublimetext.com/
[20] https://code.visualstudio.com/
[21] http://www.vim.org/
[22] http://testwisely.com/testwise

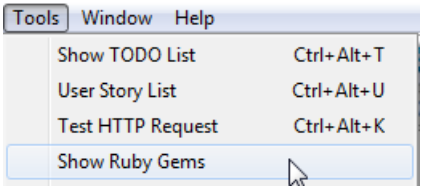


During developing (or debugging) tests, we just want to run a specific test case more commonly.



Install new gems in TestWise 5

Select menu Tools -> “Show Ruby Gems”



Type the name of the gem, click “Install” button.