How Can I Automate Stuff?

A guide for the beginner on making their testing life easier

Alan Ark
Software Development Engineering in Test - Senior
Eid Passport
aark@eidpassport.com
http://www.linkedin.com/in/arkie

Abstract

Automated testing is a hot topic right now in the QA space. But how do you get started in it? The answer is one step at a time. This talk is aimed at the beginner and hopes to provide some background in how to start to learn how to automate stuff. Stuff that can make your job easier.

Even if you **think** you don't have any automation experience, you probably actually have something that you can build on. This paper gives you one path as a guide, but ultimately it will be up to you to complete the journey.

Biography

Alan Ark *is a Software Development Engineering in Test (SDET) - Senior at Eid Passport, in Hillsboro, Oregon. Alan has gained tremendous experience working for Viewpoint Construction Software, Compli, Unircu, Switchboard.com, and Thomson Financial – First Call. Mr. Ark has previously presented 'Euro: An Automated Solution to Currency Conversion' at Quality Week '99, 'Collaborative Quality: One Company's Recipe for Software Success' at PNSQC 2008 and 'YES! You CAN Bring Test Automation to Your Company!' at PNSQC 2011. His LinkedIn profile can be viewed at* http://www.linkedin.com/in/arkie

1. Introduction

Looking at the open jobs listings, it is pretty easy to find companies asking for test automation experience. Software Development Engineer in Test (SDET), test frameworks, Selenium, Java, .Net. These are just some of the skills listed on job descriptions. When the phrase “test automation” is used, many people think of large scale test frameworks that exercise the system under test from end-to-end. That can be hard work and it is not recommended that a novice automator think about solving that problem just yet. The question is - how do you get that experience with automation if you do not know where to start? One place to start is with the definition of what automated testing can mean.

The promise of test automation is a very broad one that is fraught with danger. Many times, managers think that automation is the silver bullet that can solve all manual testing ills that live at their companies. Others have the opposite thought and believe that all automation projects will end up as shelfware and represent a large cost with little return on investment. This paper will help those on both ends of the spectrum find the right balance to learn automation and set up the effort for success.

The idea of test automation can be expanded to include ideas that are not test cases in themselves, but things that need to be in place to support the running of tests - automated or even manual ones. Some examples include setup of test environments, clearing out log files, and creation/deletion of data. If a Windows application exists for a piece of Microsoft functionality, there usually is a way that the same functionality can be accomplished from the command line. Put a few of these commands together in a batch (Microsoft TechNet website) or powershell script (Script Center website) and you have created automation!

Anything that you do manually over and over again might be a candidate for automation. If you can automate that process, it will free you up to finish other tasks. And this is where our journey begins. This paper is aimed squarely at the beginner QA engineer, who does not have a programming background, but is looking to get more involved with automation. One of the goals of this paper is to give you some ideas on how to learn the skills needed for automation (not just test automation!). Another goal is to give you some leads on further learning on your own.

The section titles come from song titles by the Beatles. Just as this talk is aimed at the beginner, many musical artists took their cue from the Beatles at one point.

2. "Don’t Let Me Down"

How some managers view automation efforts can be summed up in one word – disappointment. Some automation projects have failed to the point that all automation efforts are viewed in a negative light. To prevent this from happening, we need to show that failure is not always the case. They need to know that not all automation efforts are bad. With the right scope and design, small efforts can show great gains in productivity – the ability to work smarter, not harder. The creation of automation can be a difficult thing to achieve when you have no experience in automation to begin with. The idea is to pick the right project to start off with, to have it (and you) succeed, and show that you will not let your manager down.

Learning automation is a skill that does not come overnight. As with all learning activities, you get out only what you put in. You might have to do some research outside of office hours. You might have to read books, blogs, and white papers. You will have to talk to other people for ideas and reviews. You might have to work outside of your comfort zone. You may think that learning how to automate is hard. You may think that you cannot do it. You may think that no one will help you. Don’t let yourself down by dwelling on these thoughts. Learning how to automate is a challenge, but like other challenges, you can rise to it and learn from it. While it is true that automation is not for everyone, most people can show some level of success using the ideas that they learn from the research that they have done.

While the goal of this paper is to inspire questions and thoughts about how to learn automation skills, there are a few items that are beyond the scope of this paper. These are placed here as to not let you down.

- Build a test automation framework
- Learn how to program
- Teach you how to use Selenium. (SeleniumHQ webpage)

3. "Come Together"

What is the plan? People like to hear what has worked for others. This provides validation that someone else has achieved some level of success with their approach. The real question that needs to be answered is “What are MY pain points?” Only by learning what YOUR pain points are, can a solution that works for your situation be crafted. Only then will the plan come together and the solution start to sing.

This is the beginning of a software development project. Don’t focus on what tool or technology you think you need first. This is custom automation outside of any framework whose sole job is to get some “stuff” done. You should not feel that the solution needs to be wedded to a particular tool at this point. Let the exploration of the problem lead to the tool or tools to be used.

First, think about the problem space. What are some activities that are done over and over again? Which of those activities take little time to do, but are error prone? Those are your first candidates for automation! The smaller the perceived scope of the task, the higher the likelihood of success for the automation project. Examples of small automation tasks like this would be

- Copy files from one location to another
- Backing up log files and clearing out the old ones.
- Modifying a configuration file
- Stopping and restarting services
- Modifying registry settings.

While these activities in themselves are not a test automation project per se, they indeed are steps that are needed to get a testing environment set-up. Since they need to be done anyways, they make GREAT candidates for automation, and will help you learn how to automate in the long term.

Once you identify an activity for automation, you need to define a clear goal. Defining a goal is a very important step to any software development project. This is the purpose of choosing a small activity to start off with. The activity should be of such a small scope that by identifying what the task is, the goal is perfectly described by the definition of the task. The examples listed above should give you an idea of the size of the problem to begin automation with. Once the goal has been defined, keep sight of this goal. Don’t be distracted by shiny objects that might take you off course. By focusing at the task at hand, you will have a better chance of completing what you had originally started off doing.

Now you need to work on the design. Every software project should have some design to it. Your projects will as well. The purpose of choosing an activity that you have done over and over again is that you will know exactly the steps that are needed. If you know exactly which steps are needed to finish the task, it will immediately help with the design of the solution. If nothing else, the design of the automation could be to implement each step that you would have to do manually. Believe it or not, that is an acceptable design for automation projects! Simpler really is better when you are first starting out. Don’t make things more complicated than they need to be.

When you are successful with very atomic tasks, the next step would be to string a few of the tasks together to form something larger. A great example of this is pushing out a build to a web server. A batch script can be created to do the following (familiar) things:

- Stop Internet Information Services
- Clean out directories
- Push out the new files to the proper location
- Reregister the libraries used by your product
- Copy the correct configuration files to the correct directories
- Start Internet Information Service (IIS) on your servers.

The net effect of putting these tasks together in the right order is that you have now automated a one-step deploy of code to a web server under test. This is automation at work!

4. "Get Back"

With larger automation projects, simply identifying the steps that are taken will not translate directly into automation design. In these cases, a little more thought will be needed. One common method to coming up with a workable design is to breakdown the problem into a set of smaller problems that are a bit easier to solve - the divide and conquer approach. If it turns out that you are having trouble using this approach to tackle the candidate for automation, then it's a red flag that maybe you need to start with a different project first. If you find these red flags popping up, then maybe you need to take a step back and get back on track.

Red flags raised during the goal definition and design process are valuable pieces of information. They might identify that the project you have chosen is not quite the right one to try to implement at this moment in time. Knowing when you might be over your head is a valuable skill to have. It will save both you and your manger time and money. Remember when some managers thought that all automation projects were doomed to failure? Not knowing when the staff was in over their heads could have been a reason for that failure. This is something to avoid. A good technique to address this issue is the time box. The time box should force you to think creatively and enlist the use of your peers to come up with a solution and also prevent you from spending days or weeks on any one particular issue.

Another thing to avoid, are automation activities that are just hard to automate. An example of something that is difficult to automate would be an operating system install on a bare metal machine. Another example would be an activity that is done infrequently enough that running the steps by hand would take much less time than creating automation for it. Some things are not meant to be automated. The threshold for this discussion would really depend on the situation it comes up in. It is a formula that takes into account the people involved, the business value, and the perceived risk among many other facets. There is no *magic* formula that works in all situations, but there is certainly guidelines to be followed by your company.

What do **YOU** want to try?

5. "Help!"

Everyone gets stuck somewhere along the way when creating software. The best thing that you can do is get unstuck quickly. How do you do this? Ask for help from people you know. Asking for help is not a bad thing. Believe it or not, your peers are more than willing to help you learn if you ask for their help.

Don't forget to let Google be your friend. Search on keywords to the problem that you are stumped on. It should lead you to message boards and blog posts that open up more doors to exploration. Maybe it will lead to alternative solutions that you did not think about. Also, don't expect that all of things that you try will work. As with most software activities, there is usually more than one solution that could be used. Don't be afraid to try something new. Sometimes, investigating a different solution will be the beginning of the solution that you implement.

As mentioned in a previous section, don't try to do anything fancy. Simpler is better in most cases. Also, don't try to solve all the world's problems in one swoop. Divide and conquer! Even if the solution cannot be totally automated, maybe you can get most of it automated and have some manual steps to finish the process off. That could be thought of as a form of success. At least *some* of the work is now automated. Now you'll gain more time to work on other tasks. Build on these small victories and celebrate them! By building on these victories, you will soon have a portfolio of success that you can build on.

You will start to recognize patterns, and reuse some of the ideas from your past in new projects in the future. Leverage those past successes into future solutions. You should find yourself using some code snippets over and over again. This is a good thing! Recognize this pattern early so that you can be secure that the solutions to some tasks already live in your toolbox, ready to be deployed at will! Also by showing success in your endeavors, you gain more trust from your peers and management.

Learning is a process where you see what works and what doesn't. Each idea that you try is a part of this process. Remember both the things that went well and didn't go so well. Creative brainstorming, problem-solving skills, and experimentation can all help achieve the wanted outcome. Sometimes reframing the problem statement can help you get to the final destination.

Let's revisit the difficult automation problem of installing an operating system onto a bare metal machine. What if we reframed the question? One could use virtual machines as a part of the solution. You still need to manually setup a machine once per OS/software install combination to create the base virtual image, but now you could clone them to make an army of machines! This leads to saved effort down the road.

6. "Old Brown Shoe"

Until now, this paper has shied away from touting any language over another for the novice automator. There are lots of code examples out on the Internet that people can peruse to examine how to connect to SQL servers, drive a browser, or interact with a CSV file. Feel free to use any language or tools that will help you move closer to your end game. This section will showcase how to do these things with Ruby (Ruby – A Programmer's Best Friend website). Ruby is a comfortable old brown shoe that can be used when one needs to get things done. There are other options such as Python (Python Programming Language website) and powershell that could be used as well.

In the opinion of this author, one of the strengths of using Ruby is in that it is platform agnostic. It is available for free on just about any platform. No license needed! It is an interpreted language which usually means its fast to edit and run programs. You can use the Interactive Ruby Shell (irb) and start investigating the language and its libraries (called gems in Ruby-ese). It is also open source with a strong community of contributors.

Another great "feature" with Ruby is that most of the gems will come with its set of unittests that are run to ensure that the libraries are working properly. In other words, the gems come with a set of code examples that you can run! You can use those tests to see the code in action and try to use it to suit your own needs. Be sure not to blindly cut and paste code from these tests (or any other examples) without understanding what that code is doing. If for some reason, there is a bug in the code that you pasted, you better know how to fix the problem!

The following Ruby examples can be cut and pasted into irb and used as the basis of your experimentation. You will need to install the following gems

- Watir-Webdriver (WW) (Watir-Webdriver website)
- Database Independent Interface (dbi) (Tutorialspoint – simple easy learning website)
- Database Driver – Open Database Connectivity (dbd-odbc) (ODBC bindings for Ruby website)

Pointers to more information for the above gems are included in the references. There is also a link to a WW tutorial (Ruby – A Programmer's Best Friend website) contained within the references. The WW example page referenced in the below code is a good place to start to familiarize yourself with some of the things that WW can do, and because WW is built upon Ruby, you have the entire Ruby interface available to use. The possibilities are endless. Each possibility should share something in common – judicious use of comments. It is the comments in the code that will make it easier for someone (perhaps you!) maintain the code in the future. Without comments, your code will be much more difficult to manipulate in the future.

To use the SQL example, you will need to use a custom connection string to match your database and login credentials. You may want to modify the actual SQL query to something that would return results from your database as well.

To use the CSV example, you should have it point to a CSV file that contains a header row with at least two columns – 'TestName' and 'URL'. Or feel free to modify the code to use the headers in your own CSV file.

You may wish to enter commands into irb one line at a time. This way you can experiment with syntax and see the results of your handiwork right after you hit the <ENTER> key!

```
# The hash mark at the beginning of a line marks the line as a comment.
# This script uses the following libraries – called gems in Ruby-ese

# For the web example, keep these prerequisites in mind
# Be sure to have Firefox installed.
# For chrome support install chromedriver as well.
# For IE support install IEDriver as well.
require 'Watir-Webdriver'

# For the database examples, keep up to date with the Ruby ODBC
# documentation for any prerequisites that are needed.
require 'dbi'
require 'odbc'

#  The CSV gem is part of the core package.  There is no need to install this
as a separate gem.
require 'CSV'

#---- Use WW to fill in a web form
# Using a browser developer tool to investigate your web control is very
useful here
# The web page pointed to here is actually part of the Watir tutorial.
#   It is the watir example page to experiment with different input controls.
# Go find a text field by its name, and set it's value
browser = Watir::Browser.new
browser.goto("http://bit.ly/watir-example")
browser.text_field(:name => "entry.0.single").set "Watir"

#---- Use Ruby to interact with SQL Databases
# Create the connection string
# Connect to the database and execute the SQL statement
# Iterate over each result to print out the first column – which should be
the company name in this case
# Close out the SQL connection
# NOTE:  The actual SQL to be used is dependent on your schema.
```

```
sConnect = 'DBI:ODBC:Driver={SQL
Server};Server=MyServer;Database=MyDB;Trusted_Connection=yes;'
hDBI = DBI.connect(sConnect)
sSQL=hDBI.prepare("select companyname from company")
sSQL.execute()
sSQL.fetch do |row|
      # Change this statement based on your schema.
      puts 'CompanyName  --  '+ row[0]
end
sSQL.finish()

#---- Use CSV to interact with CSV files
# Use a CSV that has a header row
aList=CSV.read('myFile.csv',:headers=>true)

# Iterate thru each line in the file and print out the test case name and
target URL
# Assign each instance to a temporary variable
# For each instance, print out two columns - the name of the test and the URL
that it should run against.
aList.each {|aPage|
      # NOTE: The following statement may change based on your schema.
      sName=aPage['TestName']
      sURL=aPage['URL']
      puts sName + " - " + sURL
}
```

Pointers to the documentation for each of these libraries are contained in the References section. Other great references are the unittests that come bundled with each gem. The directory where your gem is installed should contain a directory named unittests, spec, or test. With a typical install, one may find the Watir-Webdriver unittests at C:\Ruby\lib\ruby\gems\[*ruby version*]\gems\watir-webdriver-[*gem version*]\spec, where *ruby version* and *gem version* are the respective build versions of Ruby itself, and the version of the Watir-Webdriver gem being used by the current install. The unittests are all written in Ruby. Coupling the unittests with irb should give one better insight into how to better use the gems that you are interested in.

7. "Long and Winding Road"

This is not a talk about how to program, but recognize that to gain the most from automation, you really will need to understand basic programming concepts. This is beyond the scope of this talk, but here are some concepts that you should become more familiar with when you start to look at different programing languages. Remember that learning how to program is a long and winding road. Topics for further research include:

- Data Structures
- Flow Control
- Looping constructs
- Function/Method calls
- Handing of Input, Output, and Errors.

A good resource is a cookbook. One of the best was the 'Perl Cookbook' (Perl Cookbook Wikipedia entry). Since its original publication, cookbooks for other languages have been published using the same recipes. Cookbooks for a multitude of languages such as ruby, Java, and C# can be easily found. They provide recipes that you can easily incorporate into your project. But remember you should understand what the code is doing before using it.

There are also more rigorous free online courses available thru the OpenCourseWare group (OpenCourseWare Consortium website). These are free classes offered by some of the top schools around the world. They cover a range of topics including non-technical categories such as history, writing, and many other subjects. These courses appear to be a promising source for additional learning.

8. Conclusion

This paper has presented some ideas to keep in mind while you are exploring and learning more about how automation can help you.

Remember that automation is more than just test automation. It is anything that can be created to make your job easier. Tasks that are done repeatedly are great candidates for being automated. The likelihood of a successful automation project is increased by recognizing tasks that are done often and are error-prone when done manually.

Have a clearly defined goal for each of your automation projects. Keen focus on the project goal will help keep the project on track. Being able to complete automation projects within a time budget will increase the likelihood that you will have the opportunity to work on other automation projects.

Break the project into bite size pieces if needed. These smaller tasks should be well bounded and have a better chance of success. Build on these successes as you build your skills. Use these opportunities for success to show that automation can work and work well for you and your company.

Use all of the on-line and off-line resources that are available to you. Using a search engine should be the first resource that is consulted, but it should not be the only one that you rely on. Your peers are a great point of contact that should be leveraged as well. They can lead to other paths of research for you to examine. They can also provide a different perspective on the problems that you are trying to solve.

Congratulations on taking the first step to learning about automation - researching it! Take the opportunity to examine a few different languages to see what works for you. Feel free to experiment with the different libraries that are available for you. Never walk away from an opportunity to learn and you will do well with automation. Hopefully these ideas will serve you well in your quest to learn how to automate stuff.

References

Places to start

Ruby community. “Ruby Programming Language.” Ruby – A Programmer’s Best Friend, http://www.ruby-lang.org/en/ (accessed August 5, 2013).

Python Software Foundation. “Python Programming Language.” Python Programming Language, http://www.python.org/ (accessed August 5, 2013).

Ruby community. “Ruby in Twenty Minutes.” Ruby – A Programmer’s Best Friend, http://www.ruby-lang.org/en/documentation/quickstart/ (accessed August 5, 2013).

Wikipedia community. “Perl Cookbook.” Wikipedia, http://en.wikipedia.org/wiki/Perl_Cookbook (accessed August 5, 2013).

Microsoft. “Scripting with Windows PowerShell.” Script Center, http://technet.microsoft.com/en-us/scriptcenter/dd742419 (accessed August 5, 2013).

Microsoft. “Command-line reference A-Z.” Microsoft TechNet, http://technet.microsoft.com/en-us/library/bb490890.aspx (accessed August 5, 2013).

OpenCourseWare Consortium. “The OpenCourseWare Consortium.” OpenCourseWare Consortium, http://www.ocwconsortium.org/ (accessed August 5, 2013).

tutorialspoint. “Ruby/DBI tutorial.” Tutorialspoint – simple easy learning, http://www.tutorialspoint.com/ruby/ruby_database_access.htm (accessed August 5, 2013).

Libraries referenced

Selenium community. “Selenium – Web Browser Automation.” SeleniumHQ, http://seleniumhq.org/ (accessed August 5, 2013).

Scott, Alister. “Watir WebDriver | the most elegant way to use webdriver with ruby.“ Watir-Webdriver, http://watirwebdriver.com/ (accessed August 5, 2013).

Werner, Christian. “ODBC bindings for Ruby.” ODBC bindings for Ruby, http://www.ch-werner.de/rubyodbc/ (accessed August 5, 2013).