Title: $, $$, $$$ Which QA tools are worth it at their respective price points – WEB API Edition

Author:
Victor La
vla@paraport.com

Biography:

Victor La is a Quality assurance Engineer who has spent the last 9 years working in the financial services industry & the last 3 years as a Quality assurance engineer and advocate at a Seattle FinTech organization. Victor strives to create operational efficiencies in his work testing: Big data, UI performance, UI/UX, Web APIs & Desktop and Web applications. He is passionate about improving process with free or cheaper alternatives & coaching the next generations of test engineers. When he is not tinkering around with various products, he enjoys cooking, traveling & enjoying life. Feel free to contact victor at vla@paraport.com.

Abstract:

Making the right tooling decisions can impact how effective you are when managing, testing and maintaining an application. Regardless of whether a tool is - open source, free or paid - all tools have their pros and cons. The roles and duties of Quality Assurance are vast and ever changing. However, there will always be a need to: manage test cases, verify information and automate workflow. But there is no need to reinvent the wheel! There are a number of tools/ venders /consultants out there that will probably satisfy the needs of your team. The true dilemma becomes: “What is right for my team?”, “Am I really getting what I need out of this?”, “How much does this cost?” Culminating to a simple question: “Worth it?” The process is simple! Compare different tools at different price points to see which tool is the most worth it to us at its given price point.

1. Our Dilemma

Concepts like continuous integration and continuous deployment become reality and achievable when there are sets of tests that can be run: On Commit, Ad Hoc, Scheduled, etc. However, in order to get to

that point everyone needs to be part of the conversation. From the developers, dev-ops, sys-ops, Software Development Engineers in Testing (SDET) to manual testers. The right tooling decisions can impact how effective a team is when managing, testing and maintaining an application. Regardless of if a tool is open source, free, or paid, all tools have their strengths and weaknesses. It's hard to know what tool is right or wrong for your organization, without having to spend precious time testing and evaluating each tool. However, is that entirely necessary? In the age of google, the internet and YouTube. There are easy ways to find whether a tool fits your needs without having to spend hours, days and weeks testing and evaluating. Chances are, as long as you are diligent and have your personalized set of criteria, a few hours of research and a few YouTube / plural sight videos later, you will emerge from your den with an idea of which tool is right for you. Your personalized criteria will help you quickly sort through the numerous number of tools, venders, consultants out there. The original dilemma of: "What is right for my team?", "Am I really getting what I need out of this?", "How much does this cost?" quickly boils down and becomes a data point in our criteria. Only tools that are able to meet the baseline should be evaluated, otherwise any time put into the effort would be for nothing. This process will turn the selection of a testing tool from a feel into an engineered science, and ultimately help us decide, is it worth it!

1. **Our Focus on Web APIs**

There are many layers that make up a full stack application. From the UX/UI used by users to the database(s) used to store information provided by the user and stored by the application. Web application programing interfaces or Web APIs act as an interface that allows anyone the ability to access features, logic or data through the use of HTTP protocol. It can be appropriate to think of APIs as pieces of software modules that enable systems to communicate, start process and initial tasks with other systems or programs. According to Forbes contributor Louis Columbus, "2017 is quickly becoming the year of the API economy. Businesses are now realizing that APIs can be used to create new business models and new opportunities for income streams." Through the use of API keys, business and entrepreneurs with an idea or program allow access to registered users and track their usage of the API as a means to bill the user while protecting the system as a whole from fraud and abuse. The more users and the more your API is used, the more in revenue your API is able to earn. ProgrammableWeb.com, one of the world's leading sources of news and information about internet-based APIs is currently reporting more than 19,910s publically accessible APIs. This does not even include APIs that are used in a private business to business (b2b) capacity. Since 2005, there has been an exponential growth in the number of APIs that are open to be used. The graph shown in exhibit 3 illustrates the level of growth seem in APIs since 2005.

1. **Why Web APIs**

The level of growth Web APIs has seen over the last decade presents quality assurance engineers (QAE) and software developers with an interesting problem - How do we going to keep all of the chief information officers (CIO) & chief technology officers(CTO) of the world happy by constantly and consistently releasing quality APIs and how are we supposed to test and maintain all of these APIs as we release them to the public to be used by the masses? There are many ways to go about testing your own APIs, such as: Unit Tests, PowerShell & manually. Luckily for us, there is no need to reinvent the wheel on how to test web APIs. Vendors and companies out there that have invested a significant amount of time and resources and are more than happy to sell you a product for you to be able to verify, test and maintain your Web APIs. So then, the question really becomes - Is it necessary to pay a company or is it to the advantage of your organization and your team to spend a little time and energy to create a self-hosted mechanism to test? Culminating to the ultimate question – Is it worth it?

1. The sea of solutions

There is a proverbial sea of solutions out on the marketplace that advertise their tools and solution can help you achieve continuous delivery and rapid deployments of your projects. With Web APIs being a mature and well understood framework and understood problem, many problems and uncertainties that usually come with a new technology have already been solved and addressed. A quick search on Google for the phrase “Testing Web APIs” yields over 482,000,000 results, with the top results coming from many popular providers and vendors that are used by many of the large fortune 500 companies of the world. A screenshot of these results can be seen in exhibit 4. Through a bit more surfing, poking and prodding of the wares, you can easily discover that for the most part, many of these vendors and products offer a free version and a subscription based version. The goal for our process it to be completely tool & process agnostic. Ideally, our criteria can be used to access thousands of tools and processes and provide a score that will help you decide whether or not a tool deserves a second glance.

1. Our Criteria

Before we set out to examine and uncover the dark secrets of these applications and vendors, it is imperative that we arm ourselves with the means to properly evaluate and rate the products in order to stay objective. In order to achieve a holistic picture of a given application we will look at five different categories and see how the product lives up to our standards by assigning it a score between one and five. (One being the WORST and 5 being the best) Our five categories are as followed:

1. Features
2. Price
3. Ease of use
4. Community Support
5. Repeatability

3.1. Feature rich

The features we look for in an application fall into two sub-categories, needs & wants. It is important to understand this distinction. An example of a need might be that the application must be compatible with Windows & IOS, where a want might be having a Chrome plugin. As many of the applications usually have some sort of tiered payment system. We will evaluate each tool’s features at its highest price point and attempt to break down the major features, both the “needs” of the application as well as the nice “wants” that the application may or may not have. Through experience and from colleges, I’ve compiled a list of needs & wants that we will look for when examining a tools features:

Needs:

Web or Console UI	**API**	**CD / CI**	**Asynchronous**
Monitoring	**Documentation**	**Windows/ Mac/ Linux**	**Repeatability**

Wants:

Easy to set up	**Multiple Language support**	**Multiple Format support**	**Plugins**
Playback	**Open Source**	**Free/ Cheap**	**Cloud**

While the list is not 100% exhaustive, it does give us a good picture of features we need and want to look for when evaluating a tool. Before beginning your search for a tool, it is highly recommend that you

compile a list of your individual needs & wants. The exercise should only take you 5 mins but will give an anchor point when doing a quick evaluation of a tool. Our scoring system will follow the below rubric:

1. Lacking in both needed features and wanted features.
2. Has a number of needed features with very little wanted features.
3. Has all needed features but only a few wanted features which may lead to the tool being cumbersome to use
4. A feature rich application of both needs and the majority of your wants
5. A feature rich application of all needs, wants & even bonuses

3.2 Price matters

While features are certainly important, so is how much we are paying for those features. Many of the vendors and the applications promise the holy grail of web API testing. From creating, administering to even automating your web API testing. However, behind the glitz and the glamor is a platform that requires a great deal of time and money to set up. While it is easy to see and evaluate the price seen on the "pricing" page of a tools website. It is much harder to calculate the amount of costs that you may incur from using a tool. Cost to set up a self-hosted server, training costs and support, at the end of the day, will affect the bottom-line of any business or organization. Our hope is to uncover all costs associated with using a tool and thus score it from 1-5 using the below criteria:

1. Costs are astronomical: Cost of tool exceeds $5,000 a year, Servers need building / maintenance / support staff & consultants needed
2. Costs are high, total costs per year will exceed $5,000
3. Costs are moderate, but totals cost per year will not exceed $5000
4. Costs are low and total cost per year will not exceed $1000
5. There are no costs, 100% free to use

3.3 It's bought, now how do I use this...

Let's attempt to look a week into the future after we've purchased the tool and signed the licensing agreement. Our goal is to look at the: "What needs to happen to get this thing working." Implementation can be difficult and time consuming and working with a new technology can be rough, especially if the tools do not follow best practices or have detailed documentation. It would be naive for us to believe that a tool will just work out of the box. There are a number of factors that should be considered at this stage. Those factors are as followed:

Factors

Physical	**Human**
Does the Hardware exists to support my tool?	**Do I have anyone to build out infrastructure if I need it?**
Does my tool require specialized components?	**Do i have someone who would be able to support this tool?**
Does our current tooling already support a Web API Testing Framework?	**Is there anyone else I will need to involve during the set up?**

Many of the factors above all have a common theme, which is, who am I or what am I missing in this conversation. It is important to remember, that Rome was not built in one day, nor by one man. It takes

a village, and in this case, an organization to: set up, use and perfect a tool. Take 5 mins to ask yourself the above questions. Your answers should then be evaluated against the below scale:

1. I have not talked to enough people in the organization, there are 6 or more people I need to consult before I should proceed any further | infrastructure is not set up nor is there an appetite to invest in additional hardware.
2. Infrastructure will need over 50 hours of work & people will need to be hired
3. Infrastructure exists however will need some investment for scaling purposes | People in the org have had some experience with the tool
4. Infrastructure is in place with only minimal changes needed & People are on board with setting up the new technology
5. Infrastructure in place & people are already skilled at using the tool

**Example of people who you may want to speak with:

1. Developers
2. Infrastructure
3. Dev-Ops
4. Sys-Ops
5. Security

3.4 Community support

Sometimes it's just easier to run a google search in order to quickly answer a question. While documentation can be great. Sometimes it's even better to have a mature community and well documented patterns on popular knowledge repositories (E.g. stack overflow). Other community support sources we will be looking for are: Libraries, best practices guides, mods, templates, etc.
The following items will be counted as community support tools:

Official Documentation	Stack Overflow	Community FAQ	Blogs
Guides	templates	libraries	Open Source

In this 5 min exercise. Think of a common and simple question you may have when using or setting up a tool. Some example questions would be:

1. "When trying to POST app returns 500 -insert tool name"
2. "-insert tool name- continuous integration"
3. "Best way to use -insert tool name-"

Your questions should be similar in nature and should try to address any shortcomings that you've identified thus far down this process. Once your questions have been documented, perform a search on the tools community site, google , stack overflow and evaluate whether or not your question was answered and if there is any supplementary tools or processes that can take you one step further. As for the score, we will evaluate the community support based on the below criteria:

1. Searches yield little or no information
2. Searches does not provide much help or references outdated material
3. Search yields the answers that you seek after a little digging
4. Searches yield a plethora of answers & provides follow up instructions on how to proceed
5. Searches yield a plethora of answers, best practices, and even provides ways to solve it

3.5 Automatable?

In the age of Continuous integration (CI) and Continuous delivery (CD), just being able to run functional tests is not enough. Ideally, a Web API testing platform should allow a team to make quick iterations and most importantly, give the team the ability to add features and refactor without fear of breaking changes. While automation may not be important to you now, or in the near future. It is important to understand the technology is moving faster and faster. The ability for a dev to change code at a moment's notice and have those changed into production minutes later is increasingly valuable. Our criteria on how repeatable a tool is as follows:

1. The tool does not support automation and is for manual testing only
2. The tool can be used for automation, however, tests can only be performed on an ad hoc basis
3. The tool supports automation through the use of its own internal UI
4. The tool supports continuous integration with your desired CI framework
5. The tool supports CI/ CD development on numerous frameworks

4.0 Tips and Hints:

There are a number of very popular tools out there on the market place but sometimes it's easier to see if there are any tools that your organization already owns but just not utilized. For example if you use Smartbear as a service provider, it would be good for you to know that Smartbear partners with SoapUI and provides this service as part of their package of testing utilities. Other major vendors who provide testing applications as part of their contracts would be: Parasoft, Apache & Progress. So it is worth some time exploring some of your existing relationships to see if there is something that you are already paying for but not utilizing.

Scoring:

Scoring should be relatively straightforward. Every tool you evaluate will be out of 25. I would recommend any tool that scores lower than a B (80%) to be tossed aside as it would not be worth your time exploring the tool any longer. Where anything that scores 80% or greater should be evaluated at your own discretion. The scoring will look a little something like this:

Tool: Postman

Quality	**Score**
Feature Rich	X/5
Price	X/5
Ease of use	X/5
Community Support	X/5
Automation	X/5

Total: X/25

Tools:

Below are some popular tools that I've done research on in the past.

Postman	Visual Studio	Fiddler	Soap UI	REST-Assured
apigee	jMeter	Tricentis	Katalon	Karate DSL

5.0 Closing remarks:

The scoring and the evaluation methods detailed above do not have to stop at Web APIs. For the most part, the questions are agnostic enough for you to be able to slightly modify them and apply them to an array of tools. In addition, the questions should be general enough for anyone to be able to use them. From a QA manager to an individual contributor. I myself am not in a managing role, however, I have had a great deal of fun finding ways to make the lives of those around me a little easier. I encourage all to do a little homework and to think about what is important when it comes to your tooling and how you'd like to test now, tomorrow and in the future.

Happy testing

References

Columbus, Louis. "2017 Is Quickly Becoming The Year Of The API Economy." Forbes. January 29, 2017. Accessed June 18, 2018. https://www.forbes.com/sites/louiscolumbus/2017/01/29/2017-is-quickly-becoming-the-year-of-the-API-economy/#573b4f796a41.

Pw_honcho. "About ProgrammableWeb." ProgrammableWeb. May 22, 2014. Accessed August 20, 2018. https://www.programmableweb.com/about.

Exhibit 3:

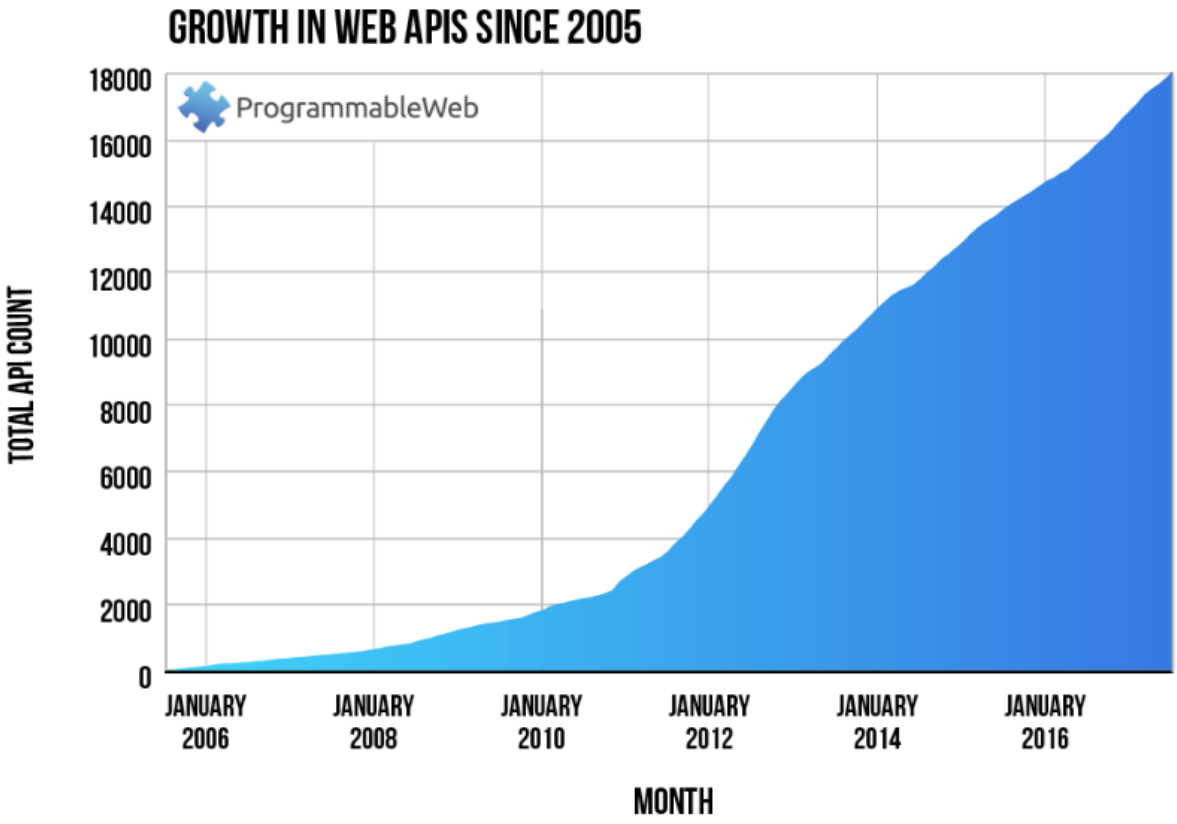

Exhibit 4: