SOFWERX

Accelerators for Web API Integration Development

Austin Alkire

Jim Ladd

August 5, 2021

Contents

Introduction

The craft of software development has evolved drastically over the past four decades. We are transitioning from blindly grasping at ideas on how to write code to creating the initial optimizations of our development processes. We don't want to history to repeat but rather to improve upon itself. We gather a set of recommendations, lessons learned, rules of thumb, etc. from each project and apply them to the next one in an endless loop.

In this paper, we present a set of recommendations with the goal of accelerating the development of integrations with web-based interfaces. These accelerators were constructed over decades of system integration experience and recently re-validated on a real-world project. While the focus of this effort was on integration development, most of the accelerators can be applied on other types of software development.

Background

System integrations have been around the software industry since the 1970's [1]. One of the first recorded technologies was the Electronic Data Interchange (EDI) which is still in use today. These technologies slowly evolved during the 1980's when every system vendor offered a proprietary interface, typically based on remote procedure calls (RPC). By the late 1990's, a wave of change flooded the architectural landscape with the rise of HTTPS/XML-based interfaces. Microsoft's Simple Object Access Protocol (SOAP) protocol was soon followed by Dr. Roy Fielding's definition of the Representation State Transfer (REST) protocol. Around this time, a web services platform, xBroker, was created using HTTPS/XML for *all* of the integration points [2]. Even with these web-based technologies making system integrations easier to design and implement, the task of interfacing with an external system remains non-trivial.

At SOFWERX, we are investigating potential integrations in order to 1) become more productive, 2) eliminate human-based errors, and 3) reduce costs. The Information Technology team was recently asked to automate the inventory management of the company's "gift shop". When a visitor purchases an item, the users want to scan a barcode on the item and then have the quantity of that item in the inventory to be decremented. This automation would eliminate the manual entry of the SKU. SOFWERX uses Square as the cloud-based point of sale system and, in order to reduce costs, decided to build the solution in-house. The high level architecture of the solution is shown below:

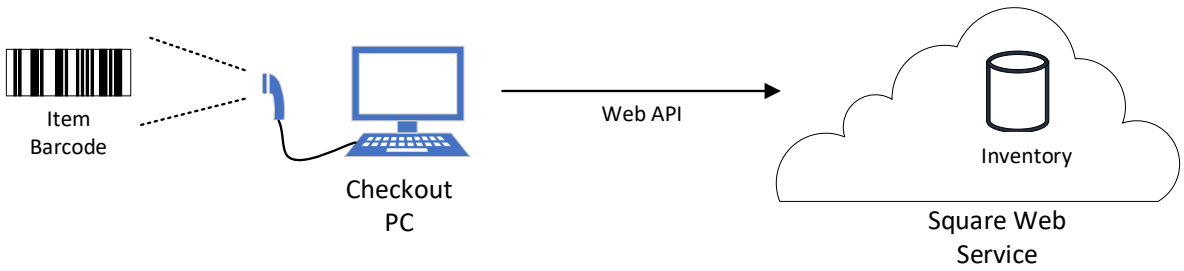

Figure 1 - High level architecture

Fortunately, but not unexpectedly, Square provides an API to their web services. After a very brief review of their documents, we expected the integration using their interface would follow this event sequence.

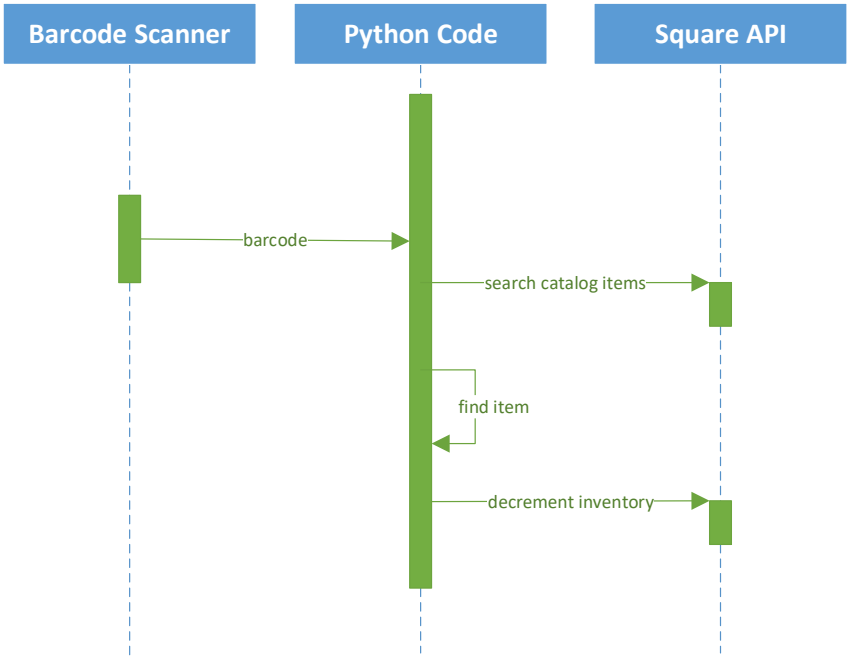

Figure 2 - Event sequence

While the problem space is not complex, it does require~~s~~ a real-world, working solution. This project was a perfect candidate to apply our accelerators for developing these types of integrations. For convenience, we refer to all interfaces using HTTPS with XML, JSON, etc. as Web Application Programming Interfaces (WAPI).

Our Top Accelerators

The IDE Is Your Best Friend

An integrated development environment (IDE) is a software application for building and testing software programs. IDEs commonly combines a code editor, compiler, debugger, and build automation tools [3]. A few advantages of an IDE include [4]:

- Serves as a single environment for most, if not all, of a developer's needs.
- Code completion capabilities improve programming workflow.
- Automatically checks for errors to ensure top quality code.
- Refactoring capabilities allow developers to make comprehensive and mistake-free renaming changes.

- Maintain a smooth development cycle.
- Increase developer efficiency and satisfaction.
- Deliver top-quality software on schedule.

While the origins of Integrated Development Environments (IDE) can be traced to the 1960s [5], they didn't become widely available until the 1980s [6]. However, in our experience at SOFWERX, IDEs are not being utilized to their full potential by college students, interns, and software developers who are early in their career. Anyone who aspires to write software as a profession should know at least the basics of how to debug code within an IDE. An IDE, any IDE, will make the task much easier and the productivity much higher. Learn it, love it, live it.

The Three Commandments

Software developers face tens, hundreds, or even thousands of decisions during the lifecycle of a project. Even on projects that are scoped to a level like an integration with a single external system, the task at hand can be daunting. A few of the reasons why writing software is so difficult include [7] [8]:

- Hardware technology improves quickly, making ever-more complex software practical and desired.
- New (and new versions of) languages, libraries, and frameworks are rapidly released.
- Programming paradigms come and go.
- Coordination and communication among team members increases with application size.
- Requests for features are often ill-defined.
- Every line of code is a potential point of failure.
- Lack of user input.
- Users don't know what they want until they see it.
- All software is affected by external factors.

To help focus the development efforts, we use *The Three Commandments of Software Development*. The directives are listed below and the sequence is critical:

1. ***Thou shall make it work***
   The first commandment is to get the code working…period. Frequently this is straight-line code even if an object-oriented language is used. It does NOT matter how ugly or crude the first version of the code is, JUST GET THE CODE WORKING! Once it is, you will have enough insight and confidence to proceed to the next commandment.

2. ***Thou shall make it fast***
   After the code works, *and only after*, focus on making the code fast *enough*. Now is the time and this is the place for researching different algorithms and designs to make the code fast *enough*. Not fast as possible, only fast *enough*.

3. ***Thou shall make it pretty***
   Once the code is working in a timely fashion, the effort should be directed toward beautification and elegance. This commandment forges the code into the production ready form.

This approach should be applied in three increments (one per commandment) with one or more iterations within each increment. The creation of unit test cases usually begins when pieces of the code become "stable". At SOFWERX, this begins during the end of the first increment and completes during the last increment. Also, our definition of stable is not metric-based but is totally subjective. It occurs when the developer becomes satisfied with his/her code.

First Manualation Then Automation

Most software programs can be casted (pun intended) as mechanisms of automation. Software eliminates or lessens tedious tasks, decreases errors related to manual activities, and increases the velocities of work execution by automating the underlying processes. Before these processes should be automated by assembling new source code, they should be manually executed. One should understand, in detail, the target process before attempting to automate it.

In the scope of this paper, the process we are attempting to automate is an integration with an external system via a WAPI interface. This process typically consists of one or more API invocations in a known and controlled sequence. We highly recommend manually executing the steps within the integration process before constructing code to automate it. The manual approach consists of:

1. Invoke the Web API
2. Receive the response
3. Review the data in the response
4. Observe the behavior of external system
5. Repeat if necessary…with real-world data, commonly from the response

We follow this process with all of the integrations. We manually execute the interface sequences before automating them via code.

Postman Is Your Second Best Friend

If the IDE is your favorite tool, Postman should be the next one in the queue. Postman is a collaboration platform for API development that simplifies each step of building an API [9]. It has a very intuitive user interface and several important features. Postman lessens the burden of integration development by providing the most popular authorization types like OAuth 2.0, Bearer Token, and AWS Signature.

We use Postman for the manual integration development as described in the previous section. The WAPI requests are constructed and the responses collected, often feeding into the next request configuration. With the integration with Square, the process consisted of two WAPI requests. The first was to query the catalog with the SKU from the barcode and the second was to decrement the inventory for that catalog item. Both of the WAPI requests were developed and then refined within Postman before migrating the knowledge to Python.

We regard Postman as absolutely critical for developing integrations with WAPI interfaces. Learn it, love it, live it.

Test Then Trust

Similar to the network security mantra of "trust but verify", we approach an interface of a new system with "test then trust" pattern. After decades of building integrations, we have observed both excepted and unexcepted responses. Similar to Edward Lorentz's foundation of chaos theory in which small changes in initial conditions produced large changes in long-term outcome [10], we have observed drastic differences in the responses of a system with a small change in the request data.

In an attempt to uncover any lurking butterflies (a nod to Lorentz's work), we perform boundary testing on the WAPIs. We create requests that will return 1) an empty result set, 2) a result set of zero items, 3) a result set of one item, 4) a result set of "many" items, and 5) a result set of the maximum number of items.

While on the topic of unexpected behavior, we ran across an issue with the Square API. At the time of our development, there were two versions available. We initially decided to use the latest version (V2) of their published interface. While the first request (i.e., search catalog items) worked with the V2 API, the second request to decrement the inventory proved to be unreliable for all of the catalog item types. After much testing and many iterations in Postman, we opened a trouble ticket with Square that led to no resolution. The final resolution was to use the first version (V1) of the API for the decrement request. The recommendation here is to test early and often and do not hesitate to mix versions of a published API.

Postman Is Really Your Best Friend

During an earlier integration project with a different system, we discovered that the authorization method was a custom method based on a temporary API key. The available documentation that explained the authorization method and related algorithm was confusing. Fortunately, the vendor provided an existing collection of Postman requests that included the JavaScript code for the authorization method. After downloading the collection and exercising the APIs for our integrations, we ported the JavaScript code to Python for our integration solution.

Earlier in this document, we lied…when it comes to Web API integrations, Postman really is your favorite tool.

Summary

In this paper, we presented a set of recommendations with the goal of accelerating the development of integrations with web-based interfaces. These accelerators were collected and improved over decades

of system integration experience and successfully re-validated on a real-world project. While the focus of this effort was on integration development, most of the accelerators could be applied on other types of software development.

Who We Are

SOFWERX is a non-profit entity that accelerates evolution of the Warfighter through technology discovery, engagement, development, and transition. SOFWERX was created under a Partnership Intermediary Agreement, established in September of 2015, between DEFENSEWERX and the United States Special Operations Command (USSOCOM).

Austin Alkire is the System Administrator Lead at SOFWERX where he leads the system administration team. Austin joined SOFWERX in 2019 after working as a volunteer and intern. He earned a BS in Information Technology from University of South Florida. Austin's LinkedIn profile link is www.linkedin.com/in/austin-alkire-a96134153

Jim Ladd is a Senior Software Architect and IT Manager at SOFWERX where he leads the software engineering, web development, and system administration teams. Jim has been developing software solutions for over 35 years. Before joining SOFWERX in 2019, Jim was CEO and Principal Consultant at Wazee Group, a niche consulting company, for 20 years. His LinkedIn profile link is https://www.linkedin.com/in/jim-ladd/

References

[1] F. Doglio, "The Evolution of Systems Integration," 30 08 2018. [Online]. Available: https://dzone.com/articles/the-evolution-of-systems-integration.

[2] J. Ladd, "xBroker - Web Services Platform," Wazee Group, Inc., 10 09 2004. [Online]. Available: http://wazeegroup.com/documents/xBroker.pdf.

[3] Red Hat, "What is an IDE?," [Online]. Available: https://www.redhat.com/en/topics/middleware/what-is-ide.

[4] Veracode, "History of IDE," [Online]. Available: https://www.veracode.com/security/integrated-development-environment.

[5] Wikipedia, "Dartmouth BASIC," [Online]. Available: https://en.wikipedia.org/wiki/Dartmouth_BASIC.

[6] A. Walker, "What is an IDE (Integrated Development Environment)?," [Online]. Available: https://www.g2.com/articles/ide.

[7] J. Lowery, "Why is Software Development Difficult?," 12 08 2018. [Online]. Available: https://levelup.gitconnected.com/why-is-software-development-difficult-81a2a041d35e.

[8] P. Smyth, "7 Reasons Why Software Development Is So Hard," 08 08 2012. [Online]. Available: https://www.finextra.com/blogposting/6836/7-reasons-why-software-development-is-so-hard.

[9] Postman, "The Postman API Platform," [Online]. Available: https://www.postman.com/api-platform/?utm_source=www&utm_medium=home_hero&utm_campaign=button.

[10] Wikipedia, "Edward Norton Lorenz," [Online]. Available: https://en.wikipedia.org/wiki/Edward_Norton_Lorenz.