AUTOMATED API TESTING

TESTING JOURNAL

AUTHOR

Anton Smirnov

2021

Automated API testing.

Learn automation testing fundamentals fast

This version was published on 2021-02-23

Contact details:

- antony.s.smirnov@gmail.com

Related Websites:

- Automated API testing: https://test-engineer.site/
- Author’s Software Testing Blog: https://test-engineer.site/

Table of Contents

Introduction.

«Software testing has become a critical and an ever-growing part of the development life-cycle. Initially, it relied on large teams executing manual test cases. This has changed in recent years as testing teams have found a way to facilitate a faster deployment cycle: test automation. A cost-effective automation testing strategy with a result-oriented approach is always a key to success in automation testing. In this book let’s see how to build good API automation scripts. »

This book is based on more than 3+ years of experience in the field of API testing automation. During this time, a huge collection of solved questions has accumulated, and the problems and difficulties characteristic of many beginners have become clearly visible. The automated testing scripts for testing API services were repeatedly created. It was obvious and reasonable for me to summarize this material in the form of a book that will help novice testers quickly build automated testing scripts for API on a project and avoid many annoying mistakes.

This book does not aim to fully disclose the entire subject area with all its nuances, so do not take it as a textbook or Handbook — for decades of development testing has accumulated such a volume of data that its formal presentation is not enough and a dozen books.

Also, reading just this one book is not enough to become a "senior automated testing engineer". Then why do we need this book?

First, this book is worth reading if you are determined to engage in automated testing – it will be useful as a "very beginner" and have some experience in automation.

Secondly, this book can and should be used as reference material.

Thirdly, this book — a kind of "map", which has links to many external sources of information (which can be useful even experienced automation engineer), as well as many examples with explanations.

This book is not intended for people with high experience in test automation. From time to time, I use a learning approach and try to “chew” all the approaches and build the stages step by step.

Some people more experienced in software test automation also having may find it slow, boring, and monotonous.

This book is intended for people who first approach the study of automation testing, especially if their goal is to add automation to their test approach.

First of all, I wrote this book for a tester with experience in the field of “manual” software testing, the purpose of which is to move to a higher level in the tester career.

Summary:

We can safely say that this book is a kind of guide for beginners in the field of automation software testing.

I have a huge knowledge of the field of test automation. I also have quite a lot of experience building automation on a project from scratch. I have repeatedly had to develop and implement the process of testing automation on projects.

The learning approach focuses on a huge chunk of theory on building the automation process. The book also discusses the theory of test automation in detail.

However, the direction of automation to support testing is no longer limited to testing, so this book is suitable for anyone who wants to improve the use of automation: managers, business analysts, users, and, of course, testers.

Testers use different approaches for testing on projects. I remember when I first started doing testing, I was drawing information from traditional books and was unnecessarily confused by some concepts that I rarely had to use. And most of the books, to my great regret, did not address the aspects and approaches to test automation. Most books on testing begin by showing how you can test a software product with basic approaches. But I do not consider the approaches and implementations of test automation at the testing stage.

In this book, I will not consider how to create and structure applications. This is useful knowledge, but it is beyond the scope of this book.

My main goal is to help you start creating automation scripts for testing API services using tools and libraries and have the basic knowledge you need to do so.

This book focuses on theory rather than a lot of additional libraries, because once you have the basics, building a library and learning how to use it becomes a matter of reading the documentation.

This book is not an "exhaustive" introduction. This is a guide to getting started in building a test automation strategy. I focused on the examples.

I argue that in order to start implementing automation scripts, you need a basic set of knowledge in testing and management to start adding value to automation projects.

In fact, when I started creating the test automation scripts, I used only the initial level of knowledge in the field of testing and management.

I also want the book to be small and accessible, so that people actually read it and apply the approaches described in it in practice.

Acknowledgments.

This book was created as a “work in progress” on **leanpub.com**. My thanks go to everyone who bought the book in its early stages, this provided the continued motivation to create something that added value, and then spends the extra time needed to add polish and readability.

I am also grateful to every QA engineer that I have worked with who took the time to explain their approach. You helped me observe what a good QA engineer does and how they work. The fact that you were good, forced me to ‘up my game’ and improve both my coding and testing skills. All mistakes in this book are my fault.

Chapter 1. Application Programming Interface.

In the world of information technology, for the Application Programming Interface, introduce the term API. API is an acronym, and it stands for “Application Programming Interface.”

Think of an API like a menu in a restaurant. The menu provides a list of dishes you can order, along with a description of each dish. When you specify what menu items you want, the restaurant’s kitchen does the work and provides you with some finished dishes. You don’t know exactly how the restaurant prepares that food, and you don’t really need to.

Similarly, an API lists a bunch of operations that developers can use, along with a description of what they do. The developer doesn’t necessarily need to know how, for example, an operating system builds and presents a “Save As” dialog box. They just need to know that it’s available for use in their app.

This isn’t a perfect metaphor, as developers may have to provide their own data to the API to get the results, so perhaps it’s more like a fancy restaurant where you can provide some of your own ingredients the kitchen will work with.

But it’s broadly accurate. APIs allow developers to save time by taking advantage of a platform’s implementation to do the nitty-gritty work. This helps reduce the number of code developers needs to create, and also helps create more consistency across apps for the same platform. APIs can control access to hardware and software resources.

API (Application Programming Interface) - a set of ready-made classes, procedures, functions, structures, and constants provided by an application (library, service) for use in external software products.

The API gives us the opportunity to use someone else's previously created interface for our own purposes. For the first time, I came across an API on the example of Windows API. This is a set of functions that any application running on this OS can use. For example, it can use standard functions to “render” the interface.

Modern APIs often take the form of web services that provide users (both people and other web services) with some information. Usually, the information exchange procedure and data transfer format are structured so that both parties know how to interact with each other.

Chapter 2. What is REST.

REST is an acronym for Representational State Transfer. It is an architectural style for distributed hypermedia systems and was first presented by Roy Fielding in 2000 in his famous dissertation.

Like any other architectural style, REST also does have its own 6 guiding constraints which must be satisfied if an interface needs to be referred to as RESTful. These principles are listed below.

Guiding Principles of REST.

1. **Client-server** – By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.

2. **Stateless** – Each request from the client to the server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.

3. **Cacheable** – Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.

4. **Uniform interface** – By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved.

In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

5. **Layered system** – The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot "see" beyond the immediate layer with which they are interacting.

6. **Code on demand (optional)** – REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented.

A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST, and DELETE data.

ARESTful API — also referred to as a RESTful web service or REST API — is based on representational state transfer (REST) technology, an architectural style, and approach to communications often used in web services development.

HTTP — Protocol of the application layer of data transfer initially — in the form of hypertext documents in the "HTML" format, is currently used for the transfer of arbitrary data.

The basis of HTTP is the "client-server" technology, that is, the existence of:

- Consumers (clients) who initiate the connection and send the request;
- Providers (servers) that are waiting for a connection to receive a request, perform the necessary actions and return a message with the result.

HTTP Protocol is designed for application and webserver to “communicate” with each other. And passed some information on the web page.

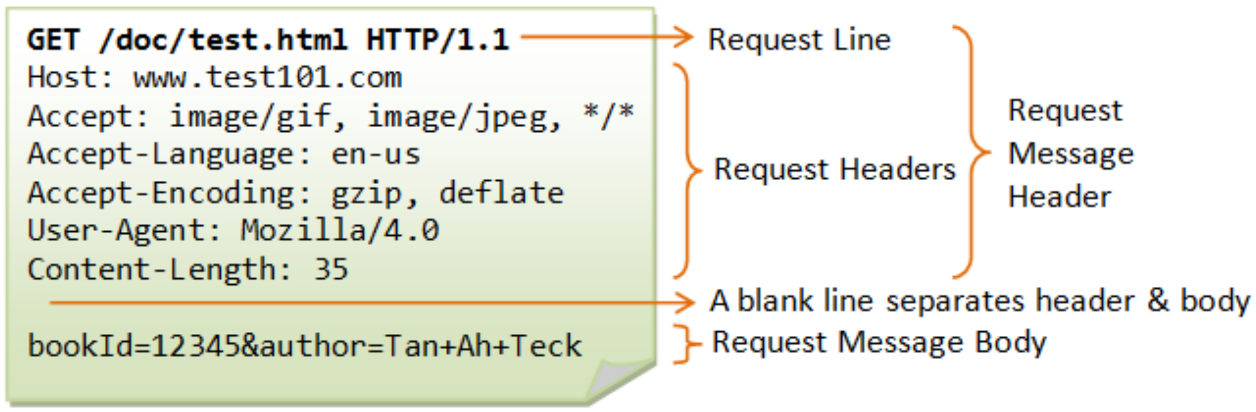
HTTP Protocol is initially Keep-Alive, what does it mean? When a user visits the site, any request is not authorized, and subsequent requests such as authorization and authentication of the user based on cookies or headers. This data is sent in a GET or POST request. Typically, authorization occurs through cookies.
A cookie is a small piece of data sent by a web server and stored on a user’s computer. A web client (usually a web browser) sends this piece of data to the web server as part of an HTTP request every time it tries to open a page on the site. It is used to store data on the user side, in practice it is usually used for:

- authenticate users;
- storage of personal preferences and user settings;
- monitor user access session status;
- maintain statistics about users.

A cookie is transmitted in any request and any server can know about who you are and what access you have based on some data, that is, based on cookies and some parameters in the GET or POST request. The most common methods for HTTP Protocol are GET and POST. The rest exist but real applications do not use them very often.

The most basic thing you need to know about any request:
When any request is sent, a packet is generated and will be sent at the application layer.

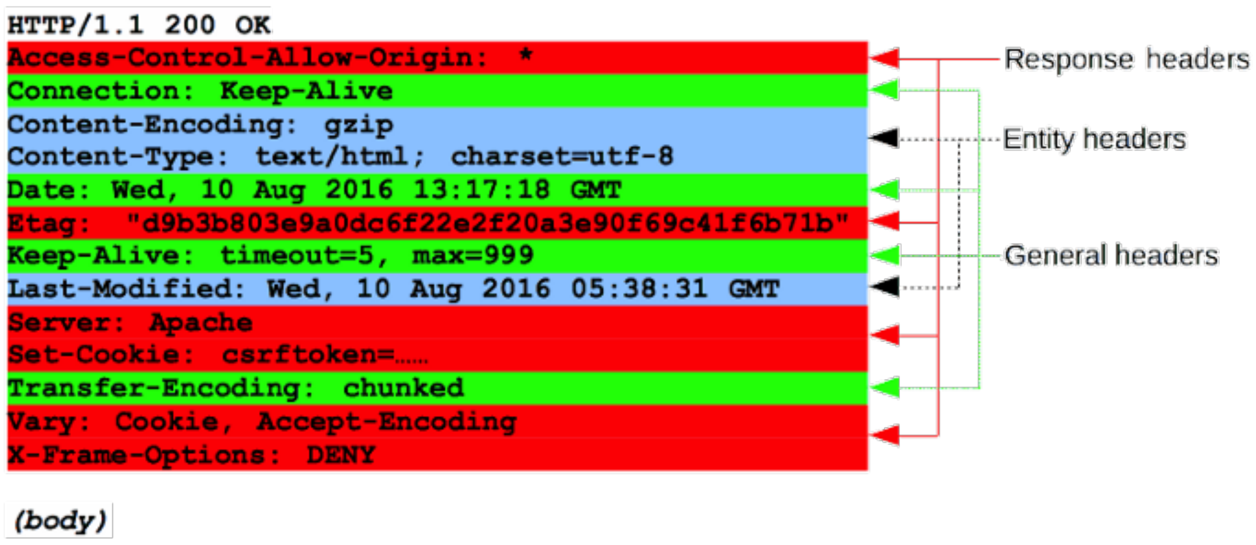
Every package consists of a request line, request header, and body.



```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

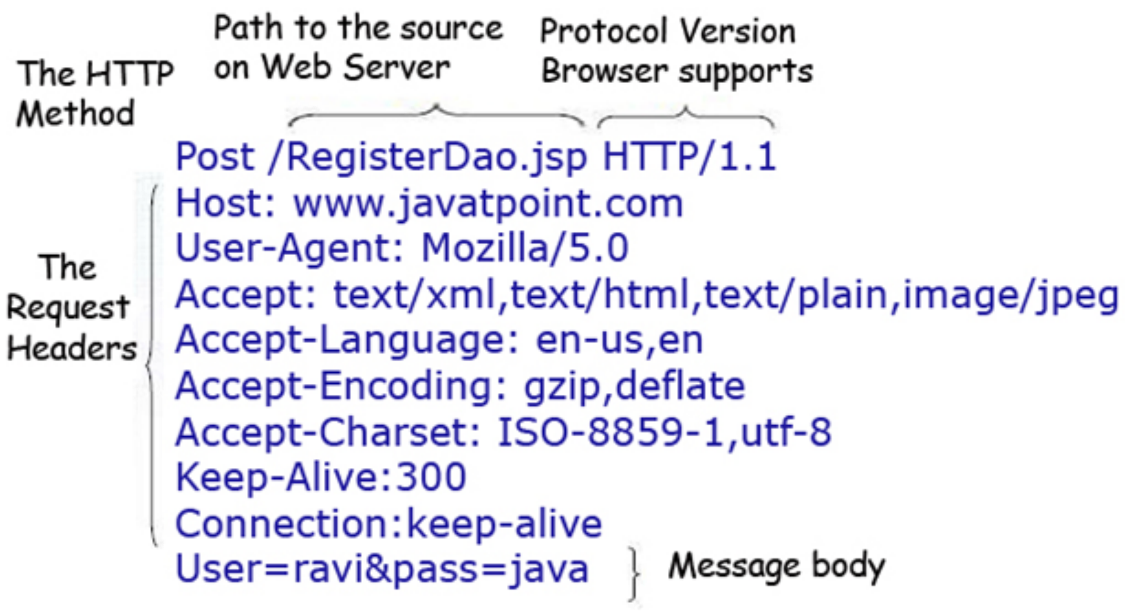
GET request sends just a header of some data. The data that is sent is located in the request message header; the data that comes in response are located in the response header.



```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 10 Aug 2016 13:17:18 GMT
Etag: "d9b3b803e9a0dc6f22e2f20a3e90f69c41f6b71b"
Keep-Alive: timeout=5, max=999
Last-Modified: Wed, 10 Aug 2016 05:38:31 GMT
Server: Apache
Set-Cookie: csrftoken=......
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding
X-Frame-Options: DENY

(body)
```

POST request sends not only request headers but also message body. In the body, you can pass any data format, JSON, XML, Media file, data, authentication and etc.


The HTTP Method
Path to the source on Web Server
Protocol Version Browser supports
The Request Headers
Message body

```
Post /RegisterDao.jsp HTTP/1.1
Host: www.javatpoint.com
User-Agent: Mozilla/5.0
Accept: text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive:300
Connection:keep-alive
User=ravi&pass=java
```

What is the difference between the GET request and POST?
The GET contains the only header, POST contains header and body. And the body contains a simple data structure.

What happens if you pass the header and body to GET? Nothing will happen. GET will work without errors.

Request Method
The request method indicates the method to be performed on the resource identified by the given Request-URI. The method is case-sensitive and should always be mentioned in uppercase. The following table lists all the supported methods in HTTP/1.1.

	Method and Description
1	GET The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.
2	HEAD Same as GET, but it transfers the status line and the header section only.
3	POST A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
4	PUT Replaces all the current representations of the target resource with the uploaded content.
5	DELETE Removes all the current representations of the target resource given by URI.
6	CONNECT Establishes a tunnel to the server identified by a given URI.
7	OPTIONS Describe the communication options for the target resource.
8	TRACE Performs a message loop back test along with the path to the target resource.