EXPERT INSIGHT

API Testing and Development with Postman

API creation, testing, debugging, and management made easy

Second Edition

Dave Westerveld

packt

API Testing and Development with Postman

Second Edition

API creation, testing, debugging, and management made easy

Dave Westerveld

‹packt›

API Testing and Development with Postman
Second Edition

Senior Publishing Product Manager: Denim Pinto
Acquisition Editor – Peer Reviews: Gaurav Gavas
Project Editor: Meenakshi Vijay
Senior Development Editor: Elliot Dallow
Copy Editor: Safis Editing
Technical Editor: Anjitha Murali
Proofreader: Safis Editing
Indexer: Pratik Shirodkar
Presentation Designer: Rajesh Shirsath
Developer Relations Marketing Executive: Vipanshu Parashar

First published: April 2020
Second edition: June 2024

Production reference: 1130624

Published by Packt Publishing Ltd.
Grosvenor House
11 St Paul's Square
Birmingham
B3 1RB, UK.

ISBN: 978-1-80461-790-8

www.packt.com

Contributors

About the author

Dave Westerveld is passionate about sharing his knowledge and expertise to help testers stay relevant in the ever-changing world of software. He has helped many software testers through his several popular video courses and has shared his expertise at conferences, online talks, and podcasts. He has also worked in the software industry for many years and in many different roles. He has worked, among other roles, as an exploratory tester, a test automation developer, and an API integrations developer.

I would like to thank my wife, Charlene, for her unwavering support of me in everything that I do. You are the rock that makes it possible for me to show up in the world in the ways that I do. You are always excited for me to share my expertise with the world. Thank you for your love and support.

About the reviewers

Christina Thalayasingam is a software engineering people manager at Northwestern Mutual with 9 years of industry experience. She holds a bachelor's degree in computer science engineering and has worked for companies such as Sysco and Dassault Systèmes®. She has a passion for testing, and beyond her daily work, she has been an active speaker at conferences and meetups such as Star East, Women in Tech Global Conference, and TestCon Europe.

Neil McCormick is a quality assurance manager who has worked in the QA space since 1998. He began API testing in the early 2000s. He helped pioneer the testing methodologies of his company, AAA, and in 2020 was promoted to his current position. He believes testers should never stop learning, exploring, and most importantly growing – both professionally and personally.

I am grateful to the author, Dave Westerveld, and the Packt Publishing team, my bosses Jerry and Kristi, my team, and too many people in the realm of family and friends to call out.

To my wife, Nandi, thank you for the weekends and nights you sacrificed for me to be able to partake in this opportunity. You smiling at my boyish excitement when I really had fun reviewing a chapter and listening to me speak endless technobabble to you has been greatly appreciated!

Join our community on Discord

Join our community's Discord space for discussions with the author and other readers:

`https://discord.com/invite/nEN6EBYPq9`

Table of Contents

Chapter 9: Running API Tests in CI with Newman 175

Chapter 10: Monitoring APIs with Postman 199

Chapter 14: API Security Testing 275

Chapter 15: Performance Testing an API 293

Preface

In a world of fast food, fast fashion, and fast-to-market strategies, does quality even matter? The pressures of the world we live in seem to push us towards taking shortcuts, even when it comes to producing high-quality software. I am doing my part to push back against that world. I think quality matters. We have enough easily broken junk in our lives. It's time for more quality.

This book is one small stake that I've put in the ground in an attempt to help the world see more high-quality software. I hope that whether you are a professional tester, or a developer looking to learn more about testing, you will be able to join me in my attempt to improve the world through quality software applications.

APIs are becoming the backbone of the internet. They help companies to communicate with each other externally and also provide the communication infrastructure for many internal pieces of modern software systems. A marriage is held together by good communication, and so it is with the internet too. Good communication between different services is important to well functioning applications. For that reason, API testing matters for producing good-quality software.

On the surface, this book is primarily about the API testing tool Postman, but I have also tried to weave in examples and teachings that will help you to to use that tool in a way that will have a real impact on quality. If you work through this book, you will gain an in-depth grasp of how Postman works, and you will also have a solid foundation in how to think about API testing in general. I want you to have more than just the ability to manipulate Postman to do what you want it to. I also want you to be able to know when and how to use it so that you can be an effective part of creating high-quality APIs.

Who this book is for

The first person I write for is myself. A lot of the ideas I talk about in this book are things I was learning myself a few years ago. In fact, Postman comes out with new capabilities so often that some of what is in this second edition are new things I learned while writing the book.

I'm always growing and learning and I love sharing what I've learned with others to help them along on their journey.

Getting started with API testing can be overwhelming, It's a huge topic and it can be intimidating to get started, so I wrote this book primarily for those software testers and developers who find themselves needing to test an API and not knowing where to start. Throughout this book I try not to assume much in-depth programming experience although an understanding of some of the basics of programming will certainly help with some sections of the book.

If you are working as a software tester and you are interested in expanding your skills into API testing, this book is certainly for you. If you are a developer who is looking to enhance your skills around testing and quality, congratulations, you are setting yourself up for a successful career! Developers that know and understand how to produce good quality software will always be in high demand. Whatever, your background, you may be able to skim through some parts of this book, but if you spend some time with it, you will find that you come away knowing how to use Postman and how to design and write good API tests.

What this book covers

Chapter 1, *API Terminology and Types*, gets you started with some of the basic API terminology and introduces you to the different types of APIs.

Chapter 2, *API Documentation and Design*, covers the design principles that apply to creating and testing APIs, and both *how* and *why* to create useful documentation.

Chapter 3, *Open API and API Specifications*, explains what API specifications are and how to get started with using them in Postman.

Chapter 4, *Considerations for Good API Test Automation*, teaches you how to create and execute valuable and long-lasting API tests in Postman.

Chapter 5, *Understanding Authorization Options*, walks through how to use many of the API authorization methods available in Postman.

Chapter 6, *Creating Test Validation Scripts*, explains how to create and use test scripts in Postman.

Chapter 7, *Data-Driven Testing*, discusses what data-driven testing is and how to use it to create scalable tests in Postman.

Chapter 8, *Workflow Testing*, explains what workflow tests are and how to create flows in Postman.

Chapter 9, *Running API Tests in CI with Newman*, shows how to run Postman API tests at the command line with the Newman runner.

Chapter 10, *Monitoring APIs with Postman*, explains how to monitor product usage of APIs with Postman monitoring.

Chapter 11, *Testing an Existing API*, works through a hands-on example that shows what kind of tests to create when testing an existing API.

Chapter 12, *Creating and Using Mock Servers in Postman*, explains what mock servers are and how to set up and use them in Postman.

Chapter 13, *Using Contract Testing to Verify an API*, discusses what contract testing is and shows how to create and use contract testing in Postman.

Chapter 14, *API Security Testing*, gives a brief introduction to security testing and gives an example of setting up fuzz testing in Postman.

Chapter 15, *Performance Testing an API*, explains the different types of performance testing and walks through some of the features in Postman that can be used to assess API performance.

To get the most out of this book

This book is intended to equip you with skills that you can use immediately in your work as a tester or developer. If you want to get the most value that you can out of this book, put the things that you learn into practice as soon as you possibly can. Work through all the exercises in this book, but also try to take the ideas that you learn and put them into practice in the "real world" as well.

This book does not assume a lot of prior knowledge of APIs, or even development and testing principles. As long as you have a basic grasp of web technology and what software development looks like in general, you should be able to follow along with this book and pick up everything that you need. Some of the test scripts in Postman use Javascript, but you don't need to know much about how that works in order to follow along, although a basic understanding would be helpful. There are examples and challenges throughout the book. They are an important part of the book and in order to get the most out of it, you should take the time to work through them.

Download the example code files

The code bundle for the book is hosted on GitHub at `https://github.com/PacktPublishing/API-Testing-and-Development-with-Postman-Second-Edition`. We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `https://packt.link/gbp/9781804617908`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. For example: "The `/product` endpoint gives information about the products accessed by this API."

A block of code is set as follows:

```
openapi: 3.0.1
info:
    title: ToDo List API
    description: Manages ToDo list Tasks
    version: "1.0"
servers:
    -url: https://localhost:5000/todolist/api
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
/carts:
    post:
    get:
        queryParameter:
        username:
    /{cartId}:
        get:
        put:
```

Any command-line input or output is written as follows:

```
npm install -g newman
```

Bold: Indicates a new term, an important word, or words that you see on the screen. For instance, words in menus or dialog boxes appear in the text like this. For example: "Click on the **Import** button and choose the **OpenAPI** option."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: Email `feedback@packtpub.com` and mention the book's title in the subject of your message. If you have questions about any aspect of this book, please email us at `questions@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you reported this to us. Please visit `http://www.packtpub.com/submit-errata`, click **Submit Errata**, and fill in the form.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packtpub.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `http://authors.packtpub.com`.

Share your thoughts

Once you've read *API Testing and Development with Postman*, *Second Edition,* we'd love to hear your thoughts! Please `click here to go straight to the Amazon review page` for this book and share your feedback.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere?

Is your eBook purchase not compatible with the device of your choice?

Don’t worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your application.

The perks don’t stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily.

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below:

`https://packt.link/free-ebook/9781804617908`

2. Submit your proof of purchase.
3. That’s it! We’ll send your free PDF and other benefits to your email directly.

1

API Terminology and Types

Learning something new can feel a little like falling over the side of a ship. Everything is moving and you can barely keep your head above water. You are just starting to feel like you understand how something works and then a new piece of knowledge comes out of nowhere, and your whole world feels topsy-turvy again. Having something solid to hold on to gives you the chance to look around and figure out where you are going. This can make all the difference in the world when learning something new.

In this chapter, I want to give you that solid foundation. As with almost any specialty, API testing and development has its own terminology. There are many terms that have specialized meanings when you are working with APIs. I will be using some of those terms throughout this book, and I want to make sure that you and I share a common understanding of what they mean.

As much as possible, I will use standard definitions. Some terms, however, do not have clearly agreed-on definitions, and for those terms, I'll share how I intend to use and talk about them in this book. Be aware that as you read or listen to things on the internet (or even just interact with teammates), you may come across others who use the terms in slightly different ways.

This book is not a dictionary, so I don't intend to just write down a list of terms and their definitions. That would be boring and probably not all that instructive. Instead, I'll spend a bit of time on the theory of what an API is and how you test it. I will weave in explanations and definitions of important terminology throughout the text.

This chapter will cover the following main topics:

- What is an API?
- Types of API calls

- Installing Postman
- The structure of an API request
- Considerations for API testing
- Different types of APIs

By the end of this chapter, you will be able to use Postman to make API requests and have a good grasp of basic API terminology. You will also have the opportunity to work through an exercise that will help you cement what you are learning, allowing you to start to use these skills in your day-to-day work.

What is an API?

A 1969 NASA publication entitled *Computer Program Abstracts* contains a summary of a real-time display control program sold by IBM (only $310! Plus $36 if you want the documentation). The advertisement says that this program was designed as an operator-application programming interface – in other words, an API.

Application Programming Interfaces (APIs) have been around for about as long as computer code has. Conceptually, it is just a way for two different pieces of code (or a human and some code) to interface with each other. A class that provides certain public methods that other code can call has an API. A script that accepts certain kinds of input has an API. A driver on your computer that requires programs to call it in a certain way has an API.

However, as the internet grew, the term *API* narrowed in focus. Almost always now, when someone talks about an API, they are talking about a web API. That is the context I will use in this book. A web API takes the concept of an interface between two things and applies it to the client/server relationship that the internet is built on. In a web API, a client is on one side of the interface and sends requests, while a server (or servers) is on the other side of the interface and responds to the request.

Over time, the internet has changed and evolved, and web APIs have changed and evolved along with it. Many early web APIs were built for corporate use cases, with strict rules in place as to how the two sides of the interface could interact with each other. A type of API called the **Simple Object Access Protocol (SOAP)** was developed for this purpose. However, in the early 2000s, the web started to shift toward becoming a more consumer-based place. Some of the e-commerce sites, such as eBay and Amazon, started to publish APIs that were more public and flexible. This was followed by many social media sites, including Twitter, Facebook, and others. Many of these APIs were built using a pattern called **Representational State Transfer (REST)**.

This approach is more flexible than SOAP and is built directly on the underlying protocols of the internet.

The internet continued to change though, and as mobile applications and sites grew in popularity, so did the importance of APIs. Some companies faced challenges with the amount of data they wanted to transfer on mobile devices, so Facebook created yet another type of API called GraphQL. This type of API defines a query language that helps to reduce the amount of data that gets transferred, while also introducing a slightly more rigid structure to the API. Each of these different API types works well in some scenarios, and I will explain more about what they are later in the chapter. However, before I get into the details of each of these types of APIs, it is important to first understand some of the concepts that underpin all web API calls.

Types of API calls

Some calls to APIs can change things on the server, while others return data without changing anything. The terms **safe** and **idempotent** are used to describe the different ways that API calls can affect data. These terms might sound a bit intimidating, so in order to better understand them, let's look at an illustration that uses something we can all understand: LEGO pieces.

Imagine that there is a table with a couple of LEGO pieces on it and I'm sitting by the table. I represent an API, while the table represents a server, and the LEGO pieces represent objects. If you come along and want to interact with the LEGO, you must do so through me. In this illustration, the LEGO pieces represent objects on a server, I represent an API, and you represent a client. In picture form, it looks something like this:

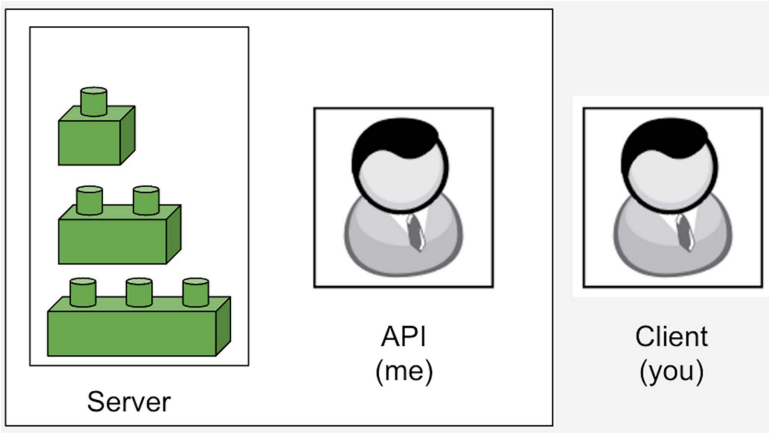

Figure 1.1: Representation of a server and a client connected by an API

You are going to be the client in this imaginary relationship. This means you can ask me to do things with the LEGO. You ask me to tell you what size the top LEGO piece is. I reply that it has a size of one. This is an example of an API request and response that is **safe**. A safe request is one that does not change anything on the server. By asking me for information about what is going on in the server, you have not changed anything on the server itself.

There are other kinds of API calls though. Imagine that you gave me a brick with a size of two and asked me to replace the top brick on the stack with the one you gave me. I do that, and in doing so I have changed the server state. The brick stack is now made up of a brick with a size of three, followed by two bricks with a size of two, as shown in the following diagram:

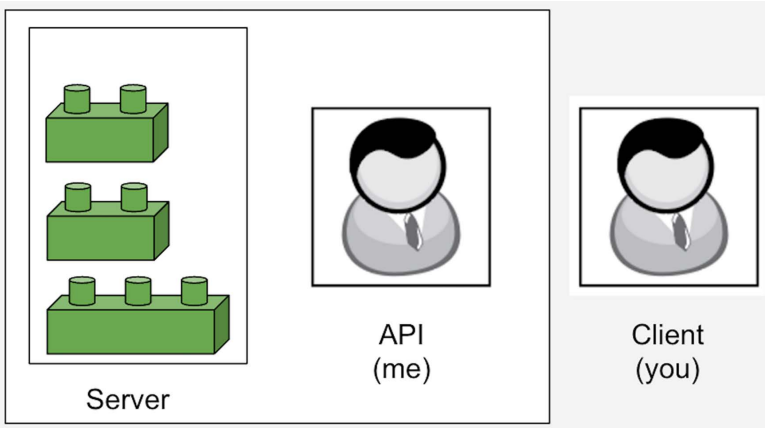

Figure 1.2: New server state

Since the server has changed, this is not a safe request. However, if you give me another brick with a size of two and ask me to once again replace the top brick with the brick you just gave me, nothing will change on the server. The stack will still be made up of a brick with a size of three followed by two bricks with a size of two. This is an example of an **idempotent** call. API calls that return the same result no matter how many times you call them are known as idempotent.

Let's imagine one more type of call. In this case, you give me a brick and ask me to add it to the top of the stack. I do that and now we have a stack of four bricks. This is clearly not a safe call, since the server has changed, but is it idempotent?

The answer is no, but take a second to think about it, and make sure you understand why this call would not be idempotent.

If you are struggling to see why it is not idempotent, think of what happens if you repeat the same request. You give me another brick and you ask me to add it to the top of the stack. If I do that a second time, is the brick stack still the same as it was after the first time you added it? No, of course not! It now has five bricks and every additional brick you give to me to add to the stack will change it. An idempotent call is one that only changes things the first time you execute it and does not make any changes on subsequent calls. Since this call changes something every time, it is not idempotent.

Safety and idempotency are important concepts to grasp, especially when it comes to testing APIs. For example, if you are testing calls that are safe, you can run tests in parallel without needing to worry about them interfering with each other. But if you are testing calls that are not safe or idempotent, you may need to be a little more careful about what kinds of tests you run and when you run them.

There are a few more important terms that we need to learn about, and we also need to dig into the structure of API requests. However, it will be a lot easier to do that if we have something concrete to look at, so at this point, let's take a brief pause to install Postman and send our first request.

Installing Postman

Postman can be run on the web or as a desktop application. The functionality and design are similar between the web and desktop applications, but in this book, I will mostly use the desktop application, so I would recommend you install it too. The app is available for Windows, Mac, and Linux, and installing it is the same as pretty much any other program you've installed. However, I would highly recommend creating a Postman account if you don't already have one. Creating an account is totally free, and it makes it a lot easier to manage and share your work. The free account of Postman is very generous in what functionality it enables. Postman does have some enterprise or advanced-level features that require a paid account, but all the examples in this book will work with the free features of Postman. However, if you don't have an account at all, it will be difficult to follow along with some of the examples, so I would strongly recommend that you register for one:

1. Go to `https://postman.com`.
2. Choose the **Sign Up for Free** option.
3. Create an account, and when asked how you want to use Postman, choose the **Download Desktop App**, and then install it as you would any program on your computer.

If you already have a Postman account but don't yet have the desktop app, you can skip steps two and three and just download it directly from the Postman home page.

I will be using the Mac version of Postman, but other than the occasional screenshot looking a bit different, everything should be the same regardless of which platform you are running Postman on.

I will primarily use the desktop application in this book, but there are times when the web application is helpful. I would recommend that you set it up as well. During the sign-up process, you can download the Desktop Agent. This agent allows the web version of Postman to get around some of the cross-origin restrictions that web browsers put on web requests. These restrictions are very important for security on the web, but when testing, we often need to be able to work around them, and this agent will allow you to do that. Once again, you can download and install this as you would any other program.

Starting Postman

Once you have Postman installed, open the application. The first time you open Postman, it will ask you to sign in. Once you've signed in, you will see the main screen with a bunch of different options for things that you can do with Postman. Don't worry about all these options right now. We will cover all of them (and more) as we go through this book. For now, just note that they are there and that you can do a lot of cool stuff with Postman. Maybe even get a little bit excited about how much you are going to learn as you go through this book!

Setting up a request in Postman

It's time to set up an API call so that we can dissect it and see how it all works:

1. To set up a new request, you can click on the + near the top of the application to add a new tab.

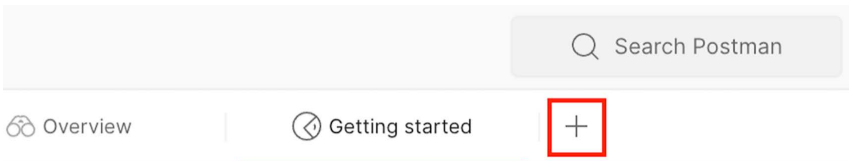

Figure 1.3: Create a new request tab

   For this first example, I will use the Postman Echo API. This is an API Postman provides that can be used to do some simple testing.

2. In the field that says **Enter URL or paste text**, type in the following URL: `https://postman-echo.com/get`.
3. Click on the **Send** button to the right of the URL field, and you should see a response from the server. Don't worry about the actual data of the response; for now, just congratulate yourself for having sent your first request with Postman!

Saving a request

Now that you have created a request, let's look at how to save requests in Postman. Postman requires that requests be saved into something called Collections. Collections are a way to collect multiple requests that belong together:

1. Click on the **Save** button above and to the right of the request URL.

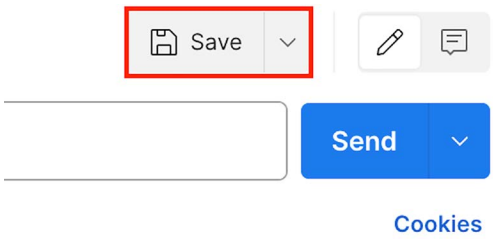

Figure 1.4: Save a request

2. In the pop-up dialog, give the request a name. Call it `Postman Echo GET`.
3. Click on the **New Collection** option at the bottom of the popup.
4. Name the collection something like `Postman Echo Requests` and click **Create**.
5. Click the **Save** button on the dialog. This will create the collection for you and save the request into that collection.

Now that you have saved a request, let's start digging into the basic structure of an API request.

The structure of an API request

The request tab in Postman provides a lot of information about the various pieces that make up an API request. Each of these pieces plays an important part in sending and receiving data with an API, so I will walk you through each one in turn. Some parts of an API request are optional, depending on what kind of request it is and what you are trying to do with it, but there are three pieces that are required for every API request. Every API request needs an endpoint, headers, and an action; let's look at those next.

API endpoints

Every web-based API request must specify an **endpoint**. In the **Postman requests** tab, you are prompted to enter the request URL. Postman asks you to enter a URL because an API endpoint is just a URL. We use the term *URL* so much that we can sometimes forget what it stands for. URL is an acronym for **Uniform Resource Locator**. The endpoint of an API call specifies the resource, or the "R" of the URL. In other words, an API endpoint is a uniform locator for a particular resource that you want to interact with on the server. URLs help you to locate resources on a server, so they are used as the endpoints in an API call.

In order to understand this, let's set up a concrete example:

1. Create a second collection by clicking on the **New** button above the navigation panel and choosing **Collection** on the popup.
2. Name the collection something like `GitHub API Requests`.
3. If you expand the collection, you will see a prompt telling you that the collection is empty, and a link reading **Add a request** is provided. Click on that link.

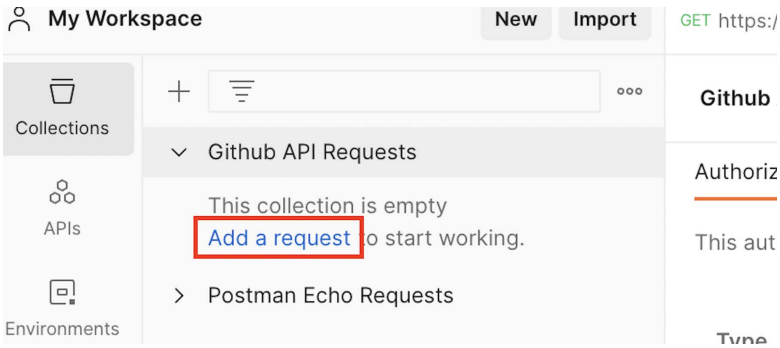

Figure 1.5: Add a request to a collection

4. Name the request `Get Repos` and fill in the request URL field with the following URL: `https://api.github.com/users/djwester/repos`.
5. Click **Send** to see the response.

   This endpoint will give you information about my public repositories on GitHub. If you have a GitHub account of your own, you can enter your username in the part of the URL where it says `djwester` and get back data for your own repositories.

You will often see an API endpoint specified without the base part of this API. So, for example, if you look at the GitHub API documentation, it will report the endpoint for this as `/users/:username/repos`. All the GitHub API calls start with the same base URL (in other words, `https://api.github.com`), so this part of the endpoint is often left out when talking about an endpoint. If you see API endpoints listed that start with a `/` instead of with `http` or `www`, just remember that you need to go and find the base API URL for the endpoint in order to call it.

API actions

Every API call needs to specify a resource that we work with. This resource is the endpoint, but there is a second thing that every API call needs. An API needs to do something with the specified resource. We specify what we want an API to do with API **actions**. These actions are sometimes called **verbs**, and they tell the API call what we expect it to do with the resource that we have given it. For some resources, only certain actions are valid, while for others, there can be multiple different valid API actions.

In Postman, you can select the desired action using the drop-down menu beside the textbox where you entered the URL. By default, Postman sets the action to **GET**, but if you click on the dropdown, you can see that there are many other actions available for API calls. Some of these actions are specialized for particular applications, so you won't run into them very often. In this book, I will only use **GET**, **POST**, **PUT**, and **DELETE**. Many APIs also use **PATCH**, **OPTIONS**, and **HEAD**, but using these is very similar to using the four that I will use, so you will be able to easily pick up on how to use them if you run into them. The rest of the actions in this list are not often used, and you will probably not encounter them much in the applications that you test and create.

The four actions (**GET**, **POST**, **PUT**, and **DELETE**) are sometimes summarized with the acronym **CRUD**. This stands for **C**reate, **R**ead, **U**pdate, and **D**elete. In an API, the **POST** action is used to create new objects, the **GET** action is used to read information about objects, the **PUT** action is used to modify (or update) existing objects, and (surprise, surprise) the **DELETE** action is used to delete objects. In practice, having an API that supports all aspects of CRUD gives you the flexibility to do almost anything you might need to, which is why these four actions are the most common ones you will see.

API actions and endpoints are required for all web APIs, but there are several other important pieces to API requests that we will consider.