24.8. Figuring Out How to Use a REST API

Suppose you have learned about the existence of an API, and want to figure out how to use it. There are five questions that you'll need to answer.

1. What is the baseurl?
2. What keys should you provide in the dictionary you pass for the `params` parameter?
3. What values should you provide associated with those keys?
4. Do you need to authenticate yourself as a licensed user of the API and, if so, how?
5. What is the structure and meaning of the data that will be provided?

The answers to these questions always depend on design choices made by the service provider who is running the server. Thus, the official documentation they provide will usually be the most helpful. It may also be helpful to find example code snippets or full URLs and responses; if you don't find that in the documentation, you may want to search for it on Google or StackOverflow.

24.8.1. Example: the datamuse API

Consider the datamuse API (https://www.datamuse.com/api/). Click on that link to open the webpage, which provides documentation. You have already seen an example of a full URL, https://api.datamuse.com/words?rel_rhy=funny (https://api.datamuse.com/words?rel_rhy=funny). Let's work through the documentation to answer the five questions.

First, in the section titled, "What is it good for?" there is a column header titled,"…use https://api.datamuse.com (https://api.datamuse.com)…". That specifies the first part of the URL: "https://api.datamuse.com/ (https://api.datamuse.com/)". However, all of the examples also include some additional characters after the `/` and before the `?`: either `words` or `sug`. These are called **endpoints**. Thus, the baseurl will be one of the two endpoints, either `https://api.datamuse.com/words` or `https://api.datamuse.com/sug`.

The answers to questions two and three, about the contents of the value of the params dictionary, can be found in the section of the documentation that describes the particular endpoint. For example, take a look at the documentation for the "words" endpoint. The entire request will return some words, and all of the params contents will specify constraints that restrict the search. If the url includes `ml=funny`, then all the words that will be returned will "have a meaning related to" to the word funny. If the url includes `rel_cns=book`, then all the words returned will have "Consonant match" to "book". It's not clear exactly what that means, but it includes words like bike and back: you can try it by visiting https://api.datamuse.com/words?rel_cns=book (https://api.datamuse.com/words?rel_cns=book)

The words to the left of the `=`, like `ml` and `rel_cns` and `rel_rhy`, will be keys in the dictionary that you pass as the value of `params` in the call to `requests.get`. The values associated with those keys will be words, like `book` and `funny`.

Many providers of APIs require you to register in advance to make use of an API, and then authenticate yourself with each request. That way they can charge money, or restrict usage in some way. A popular form of authentication is to have a personal “api_key” that you pass as one of the key=value pairs in the URL. For example, the flickr API requires that, as we will see later in this chapter (flickr.html#flickr-api-chap). Some services, such as Facebook and Twitter, require an even more complex, and secure, form of authentication, where a credential is used to cryptographically sign requests. We will not cover the use of that more complex authentication, as it is considerably harder to debug.

Currently, datamuse does not require any authentication. You can tell that because, in the section titled “Usage limits”, it states, “You can use this service without restriction and without an API key for up to 100,000 requests per day. Please be aware that beyond that limit, keyless requests may be rate-limited without notice. If you’d like to use this in a customer-facing application, or if you need a custom vocabulary, or if you plan to make more than 100,000 requests per day, you can get an API key and a stable version to use with a greater availability guarantee.”

Finally, the datamuse documentation provides a section “Interpreting the results” that explains what kind of data will be returned by the API. In this case, the structure is quite simple, it is a JSON-formatted list of dictionaries, where each dictionary provides a single word that satisfies the hard constraints in the query, and a score saying how good a match it is on the soft constraints.

24.8.2. Defining a function to make repeated invocations

Suppose you were writing a computer program that was going to automatically translate paragraphs of text into paragraphs with similar meanings but with more rhymes. You would want to contact the datamuse API repeatedly, passing different values associated with the key `rel_rhy` . Let’s make a python function to do that. You can think of it as a *wrapper* for the call to `requests.get` .

Save & Run Load History Show CodeLens

```
import requests

def get_rhymes(word):
    baseurl = "https://api.datamuse.com/words"
    params_diction = {} # Set up an empty dictionary for query parameters
    params_diction["rel_rhy"] = word
    params_diction["max"] = "3" # get at most 3 results
    resp = requests.get(baseurl, params=params_diction)
    # return the top three words
    word_ds = resp.json()
    return [d['word'] for d in word_ds]
    return resp.json() # Return a python object (a list of dictionaries in this c


print(get_rhymes("funny"))
```