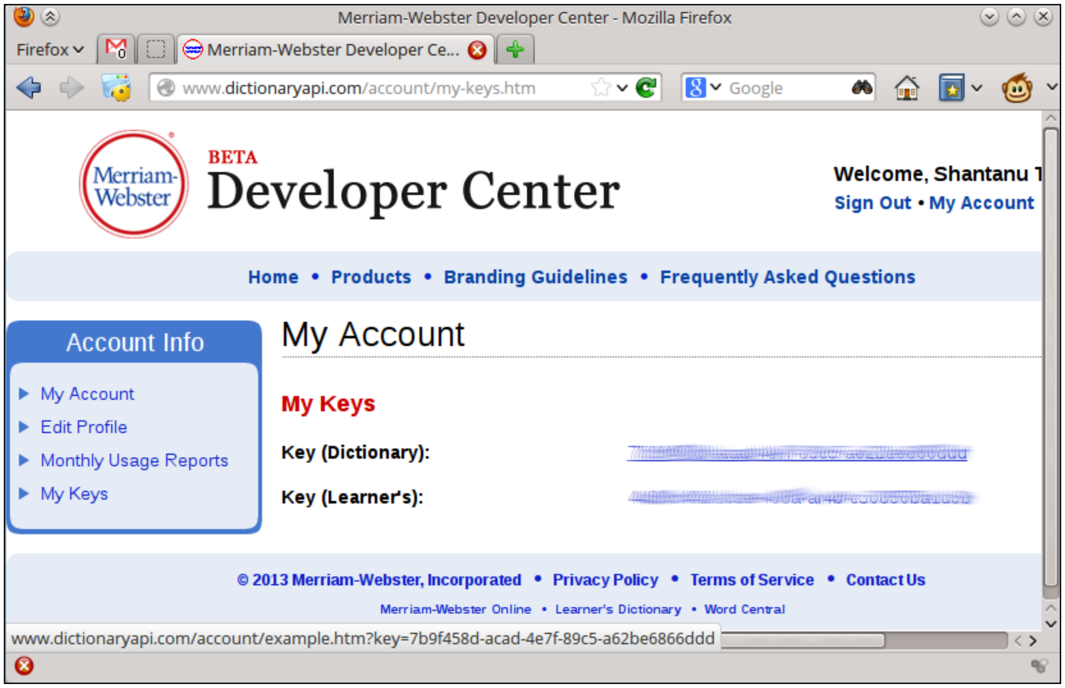
2. Log in using the newly created account and go to **My Keys** to access the keys. Note down the key for the learner's dictionary:



Merriam-Webster Developer Center - Mozilla Firefox
Firefox
Merriam-Webster Developer Ce...
www.dictionaryapi.com/account/my-keys.htm
Google
Merriam-Webster
BETA
Developer Center
Welcome, Shantanu T
Sign Out • My Account
Home • Products • Branding Guidelines • Frequently Asked Questions
Account Info
My Account
Edit Profile
Monthly Usage Reports
My Keys
© 2013 Merriam-Webster, Incorporated • Privacy Policy • Terms of Service • Contact Us
Merriam-Webster Online • Learner's Dictionary • Word Central

My Account

My Keys

Key (Dictionary): [redacted]

Key (Learner's): [redacted]

www.dictionaryapi.com/account/example.htm?key=7b9f458d-acad-4e7f-89c5-a62be6866ddd

How to do it...

Let's go through the code for the define utility script:

```
#!/bin/bash
#Filename: define.sh
#Desc: A script to fetch definitions from dictionaryapi.com

apikey=YOUR_API_KEY_HERE

if  [ $# -ne 2 ];
then
  echo -e "Usage: $0 WORD NUMBER"
  exit -1;
fi

curl --silent http://www.dictionaryapi.com/api/v1/references/learners/
xml/$1?key=$apikey | \
grep -o \<dt\>.*\</dt\> | \
sed 's$</*[a-z]*>$$g' | \
head -n $2 | nl
```

Run the script as follows:

```
$ ./define.sh usb 1
     1  :a system for connecting a computer to another device (such as a
printer, keyboard, or mouse) by using a special kind of cord a USB cable/
port USB is an abbreviation of "Universal Serial Bus."How it works...
```

How it works...

We use `curl` to fetch the data from the dictionary API webpage by specifying our API `Key ($apikey)`, and the word that we want the definition for (`$1`). The result contains definitions in the `<dt>` tags, which we select by using `grep` and then we remove the tags by using `sed`. We then pick the required number of lines from the definitions and use `nl` to put the line numbers in each line.

See also

- The *Basic sed primer* recipe in this chapter explains the `sed` command
- The *Searching and mining text inside a file with grep* recipe in this chapter explains the `grep` command

Finding broken links in a website

Some people manually check every page on a website to search for broken links. It is feasible for websites having very few pages, but gets difficult when the number of pages become large. It becomes really easy if we can automate the process of finding broken links. We can find the broken links by using HTTP manipulation tools. Let's see how to do it.

Getting ready

To identify the links and find the broken ones from the links, we can use `lynx` and `curl`. It has an option, namely `-traversal`, which will recursively visit pages on the website and build a list of all hyperlinks in the website. We can use cURL to verify each of the links for whether they're broken or not.