

AE 11: Querying the OMDB API with httr2

Suggested answers

[APPLICATION EXERCISE](#)[ANSWERS](#)

MODIFIED

October 10, 2024

Packages

We will use the following packages in this application exercise.

- **tidyverse**: For data import, wrangling, and visualization.
- **httr2**: For querying APIs.
- **jsonlite**: For some formatting

```
library(tidyverse)
library(httr2)
library(jsonlite)
```

Writing an API function

If an R package has not already been written for an application programming interface (API), you can write your own function to query the API. In this application exercise, we will write a function to query the **Open Movie Database**.

Create a request

The first step in querying an API is to create a request. A request is an object that contains the information needed to query the API. The `request()` function creates a request object. The `base_url` argument specifies the **base URL** of the API. The `req <-` syntax assigns the request object to the variable `req`.

```
omdb_req <- request(base_url = "http://www.omdbapi.com/")
omdb_req
```

<httr2_request>

GET http://www.omdbapi.com/

Body: empty

Perform a dry run

The `req_dry_run()` function performs a dry run of the request. A dry run is a test run of the request that does not actually query the API. It is useful for testing the request before actually querying the API.

```
omdb_req |>
  req_dry_run()
```

```
GET / HTTP/1.1
Host: www.omdbapi.com
User-Agent: httr2/1.0.1 r-curl/5.2.2 libcurl/8.7.1
Accept: */*
Accept-Encoding: gzip
```

Determine the shape of an API request

In order to submit a request, we need to define the **shape** of the request, or the exact URL used to submit the request. The URL of the request is the base URL of the API plus the path of the request.

APIs typically have three major components to the request:

- The **base URL** for the web service (here it is <http://www.omdbapi.com/>).
- The **resource path** which is the complete destination of the web service endpoint (OMDb API does not have a resource path).
- The **query parameters** which are the parameters passed to the web service endpoint.

In order to create your request you need to read the documentation to see exactly how these components need to be specified.

Your turn: Use the [OMDb documentation](#) to determine the shape of the request for information on *Sharknado*.

```
http://www.omdbapi.com/?apikey=your-key&t=Sharknado
```

Generate the query

Store your API key

In order to access the OMDb API you need an API key. If you do not have one, use the example key provided on Canvas.

Your turn: Store your API key in `.Renviron` so you can access it in your code. Once you have saved the file, restart your R session to ensure the new environment variable is loaded.

```
# from the console run:
usethis::edit_r_environ()
```

```
# in .Renviron add:  
omdb_key="your-key"
```

```
omdb_req |>  
  req_url_query(  
    apikey = Sys.getenv("omdb_key"),  
    t = "Sharknado"  
  ) |>  
  req_dry_run()
```

```
GET /?apikey=your-key&t=Sharknado HTTP/1.1  
Host: www.omdbapi.com  
User-Agent: httr2/1.0.1 r-curl/5.2.2 libcurl/8.7.1  
Accept: */*  
Accept-Encoding: gzip
```

Fetch the response

The `req_perform()` function fetches the response from the API. The response is stored as a response object.

```
sharknado <- omdb_req |>  
  req_url_query(  
    apikey = Sys.getenv("omdb_key"),  
    t = "Sharknado"  
  ) |>  
  req_perform()
```

```
sharknado
```

```
<httr2_response>
```

```
GET http://www.omdbapi.com/?apikey=your-key&t=Sharknado
```

```
Status: 200 OK
```

```
Content-Type: application/json
```

```
Body: In memory (893 bytes)
```

What did we get?

The HTTP response contains a number of useful pieces of information.

Status code

The **status code** is a number that indicates whether the request was successful. A status code of 200 indicates success. A status code of 400 or 500 indicates an error. `resp_status()` retrieves the numeric HTTP status code, whereas `resp_status_desc()` retrieves a brief textual description of the status code.

```
sharknado |>  
  resp_status()
```

```
[1] 200
```

```
sharknado |>  
  resp_status_desc()
```

```
[1] "OK"
```

HTTP status codes

Hopefully all you receive is a 200 code indicating the query was successful. If you get something different, the error code is useful in debugging your code and determining what (if anything) you can do to fix it

Code	Status
1xx	Informational
2xx	Success
3xx	Redirection
4xx	Client error (you did something wrong)
5xx	Server error (server did something wrong)

Tip

A more intuitive guide to HTTP status codes.

Body

The **body** of the response contains the actual data returned by the API. The body is a string of characters.

You can extract the body in various forms using the `resp_body_*` family of functions. The `resp_body_string()` function retrieves the body as a string.

```
sharknado |>  
  resp_body_string() |>
```

```
prettify()
```

```
{
  "Title": "Sharknado",
  "Year": "2013",
  "Rated": "Not Rated",
  "Released": "11 Jul 2013",
  "Runtime": "86 min",
  "Genre": "Action, Adventure, Comedy",
  "Director": "Anthony C. Ferrante",
  "Writer": "Thunder Levin",
  "Actors": "Ian Ziering, Tara Reid, John Heard",
  "Plot": "When a freak hurricane swamps Los Angeles, nature's deadliest killer rules sea,
land, and air as thousands of sharks terrorize the waterlogged populace.",
  "Language": "English",
  "Country": "United States",
  "Awards": "1 win & 2 nominations",
  "Poster": "https://m.media-
amazon.com/images/M/MV5BNTNkOTA0NjYtM2VjOC00ZmYzLWI1NzQtMjY4NTU2MGNkZmU0XkEyXkFqcGc@._V1_SX300.jpg",
  "Ratings": [
    {
      "Source": "Internet Movie Database",
      "Value": "3.3/10"
    },
    {
      "Source": "Rotten Tomatoes",
      "Value": "77%"
    }
  ],
  "Metascore": "N/A",
  "imdbRating": "3.3",
  "imdbVotes": "53,662",
  "imdbID": "tt2724064",
  "Type": "movie",
  "DVD": "N/A",
  "BoxOffice": "N/A",
  "Production": "N/A",
  "Website": "N/A",
  "Response": "True"
}
```

JSON

Here the result is actually formatted as using **JavaScript Object Notation (JSON)**, so we can use `resp_body_json()` to extract the data and store it as a list object in R.

```
sharknado |>  
  resp_body_json()
```

\$Title

[1] "Sharknado"

\$Year

[1] "2013"

\$Rated

[1] "Not Rated"

\$Released

[1] "11 Jul 2013"

\$Runtime

[1] "86 min"

\$Genre

[1] "Action, Adventure, Comedy"

\$Director

[1] "Anthony C. Ferrante"

\$Writer

[1] "Thunder Levin"

\$Actors

[1] "Ian Ziering, Tara Reid, John Heard"

\$Plot

[1] "When a freak hurricane swamps Los Angeles, nature's deadliest killer rules sea, land, and air as thousands of sharks terrorize the waterlogged populace."

\$Language

[1] "English"

\$Country

[1] "United States"

\$Awards

[1] "1 win & 2 nominations"

\$Poster

[1] "https://m.media-amazon.com/images/M/MV5BNTNkOTA0NjYtM2VjOC00ZmYzLWI1NzQtMjY4NTU2MGNkZmU0XkEyXkFqcGc@._V1_SX300.jpg"

\$Ratings

\$Ratings[[1]]

```
$Ratings[[1]]$Source  
[1] "Internet Movie Database"
```

```
$Ratings[[1]]$Value  
[1] "3.3/10"
```

```
$Ratings[[2]]  
$Ratings[[2]]$Source  
[1] "Rotten Tomatoes"
```

```
$Ratings[[2]]$Value  
[1] "77%"
```

```
$Metascore  
[1] "N/A"
```

```
$imdbRating  
[1] "3.3"
```

```
$imdbVotes  
[1] "53,662"
```

```
$imdbID  
[1] "tt2724064"
```

```
$Type  
[1] "movie"
```

```
$DVD  
[1] "N/A"
```

```
$BoxOffice  
[1] "N/A"
```

```
$Production  
[1] "N/A"
```

```
$Website  
[1] "N/A"
```

```
$Response  
[1] "True"
```

Convert to data frame

For data analysis purposes, we prefer that the data be stored as a data frame. The `as_tibble()` function converts the list object to a tibble.

```
sharknado |>
  resp_body_json() |>
  as_tibble()
```

A tibble: 2 × 25

```
  Title Year  Rated Released Runtime Genre Director Writer Actors Plot  Language
<chr> <chr> <chr> <chr>    <chr>  <chr> <chr>    <chr> <chr> <chr> <chr>
1 Shar... 2013  Not ... 11 Jul ... 86 min Acti... Anthony... Thund... Ian Z... When... English
2 Shar... 2013  Not ... 11 Jul ... 86 min Acti... Anthony... Thund... Ian Z... When... English
# i 14 more variables: Country <chr>, Awards <chr>, Poster <chr>,
#   Ratings <list>, Metascore <chr>, imdbRating <chr>, imdbVotes <chr>,
#   imdbID <chr>, Type <chr>, DVD <chr>, BoxOffice <chr>, Production <chr>,
#   Website <chr>, Response <chr>
```

Write a function to query the OMDb API

Sharknado proved so popular that four sequels were made. Let's write a function to query the OMDb API for information on any of the *Sharknado* films.

Your turn: Your function should:

- Take a single argument (the title of the film)
- Print a message using the `message()` function to track progress
- Use **throttling** to ensure we do not overload the server and exceed any rate limits. Add `req_throttle()` to the request pipeline to limit the rate to 15 requests per minute.
- Return a tibble with the information from the API

```
omdb_api <- function(title) {
  # print a message to track progress
  message(str_glue("Scraping {title}..."))

  # define request
  req <- request(base_url = "http://www.omdbapi.com/") |>
    # throttle to avoid overloading server
    req_throttle(rate = 15 / 60) |>
    # create query
    req_url_query(
      apikey = Sys.getenv("omdb_key"),
      t = title
    )

  # perform request
  req_results <- req |>
    req_perform()
```



```
# extract results
req_df <- req_results |>
  resp_body_json() |>
  as_tibble()

return(req_df)
}
```

Once you have written your function, test it out by querying the API for information on “Sharknado”. Then apply an iterative operation to query the API for information on all five *Sharknado* films and store it in a single data frame.

```
# test function
omdb_api(title = "Sharknado")
```

Scraping Sharknado...

```
# A tibble: 2 × 25
  Title Year Rated Released Runtime Genre Director Writer Actors Plot Language
<chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
1 Shar... 2013 Not ... 11 Jul ... 86 min Acti... Anthony... Thund... Ian Z... When... English
2 Shar... 2013 Not ... 11 Jul ... 86 min Acti... Anthony... Thund... Ian Z... When... English
# i 14 more variables: Country <chr>, Awards <chr>, Poster <chr>,
# Ratings <list>, Metascore <chr>, imdbRating <chr>, imdbVotes <chr>,
# imdbID <chr>, Type <chr>, DVD <chr>, BoxOffice <chr>, Production <chr>,
# Website <chr>, Response <chr>
```

```
# titles of films
sharknados <- c(
  "Sharknado", "Sharknado 2", "Sharknado 3",
  "Sharknado 4", "Sharknado 5"
)

# iterate over titles and query API
sharknados_df <- map(.x = sharknados, .f = omdb_api) |>
  list_rbind()
```

Scraping Sharknado...

Waiting 4s for throttling delay ██████████

Waiting 4s for throttling delay ████████████████████

Waiting 4s for throttling delay ██


Scraping Sharknado 2...

Waiting 4s for throttling delay ██████████

Waiting 4s for throttling delay ██


Waiting 4s for throttling delay 


Scraping Sharknado 3...

Waiting 4s for throttling delay 


Waiting 4s for throttling delay 


Scraping Sharknado 4...


Waiting 4s for throttling delay 

Waiting 4s for throttling delay 

Scraping Sharknado 5...

Waiting 4s for throttling delay 

Waiting 4s for throttling delay 

Waiting 4s for throttling delay 

```
sharknados_df
```

```
# A tibble: 10 × 25
```

	Title	Year	Rated	Released	Runtime	Genre	Director	Writer	Actors	Plot
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
1	Sharknado	2013	Not ...	11 Jul ...	86 min	Acti...	Anthony...	Thund...	Ian Z...	When...
2	Sharknado	2013	Not ...	11 Jul ...	86 min	Acti...	Anthony...	Thund...	Ian Z...	When...
3	Sharknado 2:...	2014	TV-14	30 Jul ...	95 min	Acti...	Anthony...	Thund...	Ian Z...	Fin ...
4	Sharknado 2:...	2014	TV-14	30 Jul ...	95 min	Acti...	Anthony...	Thund...	Ian Z...	Fin ...
5	Sharknado 3:...	2015	TV-14	22 Jul ...	93 min	Acti...	Anthony...	Thund...	Ian Z...	A mo...
6	Sharknado 3:...	2015	TV-14	22 Jul ...	93 min	Acti...	Anthony...	Thund...	Ian Z...	A mo...
7	Sharknado 4:...	2016	TV-14	31 Jul ...	95 min	Acti...	Anthony...	Thund...	Ian Z...	Fin,...
8	Sharknado 4:...	2016	TV-14	31 Jul ...	95 min	Acti...	Anthony...	Thund...	Ian Z...	Fin,...
9	Sharknado 5:...	2017	TV-14	06 Aug ...	93 min	Acti...	Anthony...	Thund...	Ian Z...	With...
10	Sharknado 5:...	2017	TV-14	06 Aug ...	93 min	Acti...	Anthony...	Thund...	Ian Z...	With...

```
# i 15 more variables: Language <chr>, Country <chr>, Awards <chr>,
#   Poster <chr>, Ratings <list>, Metascore <chr>, imdbRating <chr>,
#   imdbVotes <chr>, imdbID <chr>, Type <chr>, DVD <chr>, BoxOffice <chr>,
#   Production <chr>, Website <chr>, Response <chr>
```

Acknowledgments

- These exercises draw substantially on the [httr2 vignettes](#) and [reference documentation](#).

Session information

This page is built with Quarto.

[Cookie Preferences](#)