HTTP with httr2

Hadley Wickham

Chief Scientist, Posit

December 2023

1. HTTP basics
2. httr2 basics
3. Authentication
4. Multiple requests

Intro to HTTP

An HTTP request

Method
Path
Protocol version
Status
Headers

```
GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: keep-alive
Cookie: _ga=GA1.2.46172101.1675698799
Host: www.r-project.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) ...
```

An HTTP response

Status

```
HTTP/1.1 200 OK
```

Headers

```
Content-Encoding: gzip
Content-Length: 2454
Content-Type: text/html
ETag: "192e-608ff850f105c-gzip"
```

Empty line

```

```

Body

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>R: The R Project for Statistical Computing</title>
```

httr2 basics

httr2 attempts to closely match the HTTP model

```r
# Make a request
req <- request("https://r-project.org")

# Perform a request to get a response
resp <- req_perform(req)
resp
```

You can view the request & response

```
# What will happen
req_dry_run(req)

# What did happen
req_perform(req, verbosity = 1)
req_perform(req, verbosity = 2)
req_perform(req, verbosity = 3)

# If it's someone else's code
with_verbosity(req_perform(req))
```

But you’ll mostly use httr2 to speak to web APIs

```
library(httr2)

req ← request("https://api.github.com/")
resp ← req_perform(req)

resp ▷
  resp_body_json() ▷
  str()
```

Website	**Web app**
HTML	JSON
rvest	jsonlite
GET	GET + POST + ...
Mostly public	Mostly private

Lots of options for modifying the request

- `req_headers()`
- `req_progress()`
- `req_proxy()`
- `req_timeout()`
- `req_useragent()`
- `req_url()`, `req_url_query()`, `req_template()`

Which you combine together with the pipe

```
request("https://example.com") ▷
  req_url("/download") ▷
  req_url_query(state = "TX") ▷
  req_progress() ▷
  req_timeout(10) ▷
  req_perform()
```

But you don’t have to start from scratch

```
req_tx ← request("https://example.com") ▷
  req_url("/download") ▷
  req_url_query(state = "TX") ▷
  req_progress() ▷
  req_timeout(10)

req_wa ← req_tx ▷ req_url_query(state = "WA")
```

Sometimes you’ll get errors

```
req ← request("adslkfjdsalkjfadl;kfskd")
resp ← req_perform(req)

req ← request("https://api.github.com/doesntexist")
resp ← req_perform(req)

req ← request("https://api.github.com/user")
resp ← req_perform(req)

# You can always grab with
last_response()
last_request()
```

Authentication

Let’s dive into that last case

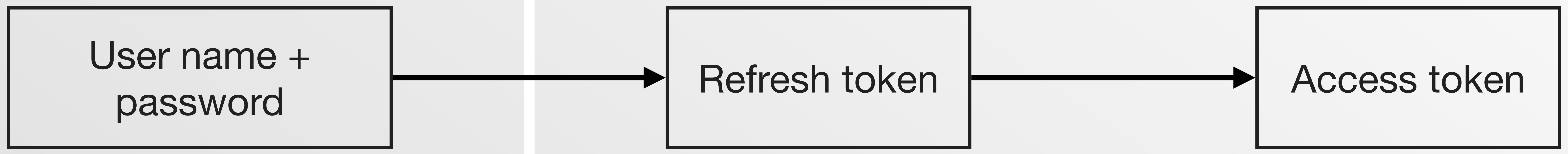
1. `req_auth_basic()`
2. `req_auth_bearer_token()`
3. `req_oauth_auth_code()`
4. `req_oauth_bearer_jwt()`
5. `req_oauth_client_credentials()`
6. `req_oauth_device()`
7. `req_oauth_password()`
8. `req_oauth_refresh()`

Let's dive into that last case

1. req_auth_basic()
2. req_auth_bearer_token()
3. **req_oauth_auth_code()**
4. req_oauth_bearer_jwt()
5. req_oauth_client_credentials()
6. **req_oauth_device()**
7. req_oauth_password()
8. req_oauth_refresh()

Demo

How does OAuth work?



Lasts until you change it; hopefully lives in a secure password store

Lasts days to months; httr2 always caches in session; can cache on disk

Lasts a couple of hours at most; sent with every request

web app

httr2

Once you’ve figured it out, wrap it in a function

```
req_auth_github ← function(req) {
  req_oauth_auth_code(
    req,
    client = example_github_client(),
    auth_url = "https://github.com/login/oauth/authorize"
  )
}
```

Multiple requests

1. `req_perform_parallel()`
2. `req_perform_sequential()`
3. $\boxed{\texttt{req\_perform\_iterative()}}$

Demo

1. Do it for one page
2. Write a function to generate the next request
3. Use req_perform_iterative() to perform it for all requests
4. Use resps_data() to join all the data together
5. Use tidyr to turn it into a data frame

Conclusion

Major differences from httr

- Can create and modify a request without performing it
- Automatically turn HTTP errors into R errors (customise with req_error())
- Much much better OAuth support
- Lots of other new features (e.g. req_retry(), req_cache(), req_throttle(), multiple requests, curl_translate())
- More thoughtful considering of package needs
- A logo!!

httr2.r-lib.org

vignette("httr2")