net/http: why do I need to read the body?

Erik Dubbelboer
github.com/erikdubbelboer

A familiar pattern

```go
func doSomeRequest() error {
    resp, err := http.DefaultClient.Get("http://example.com")
    if err != nil {
        return err
    }
    defer resp.Body.Close()

    if _, err := io.Copy(ioutil.Discard, resp.Body); err != nil {
        return err
    }

    return nil
}
```

GitHub Copilot knows about it

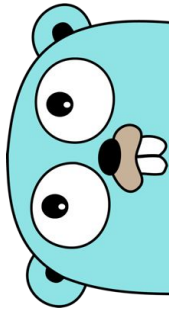
```
func doSomeRequest() error {
    resp, err := http.DefaultClient.Get("http://example.com")
    if err != nil {
        return err
    }
    defer resp.Body.Close()

    if _, err := io.Copy(ioutil.Discard, resp.Body); err != nil {
```

Why?

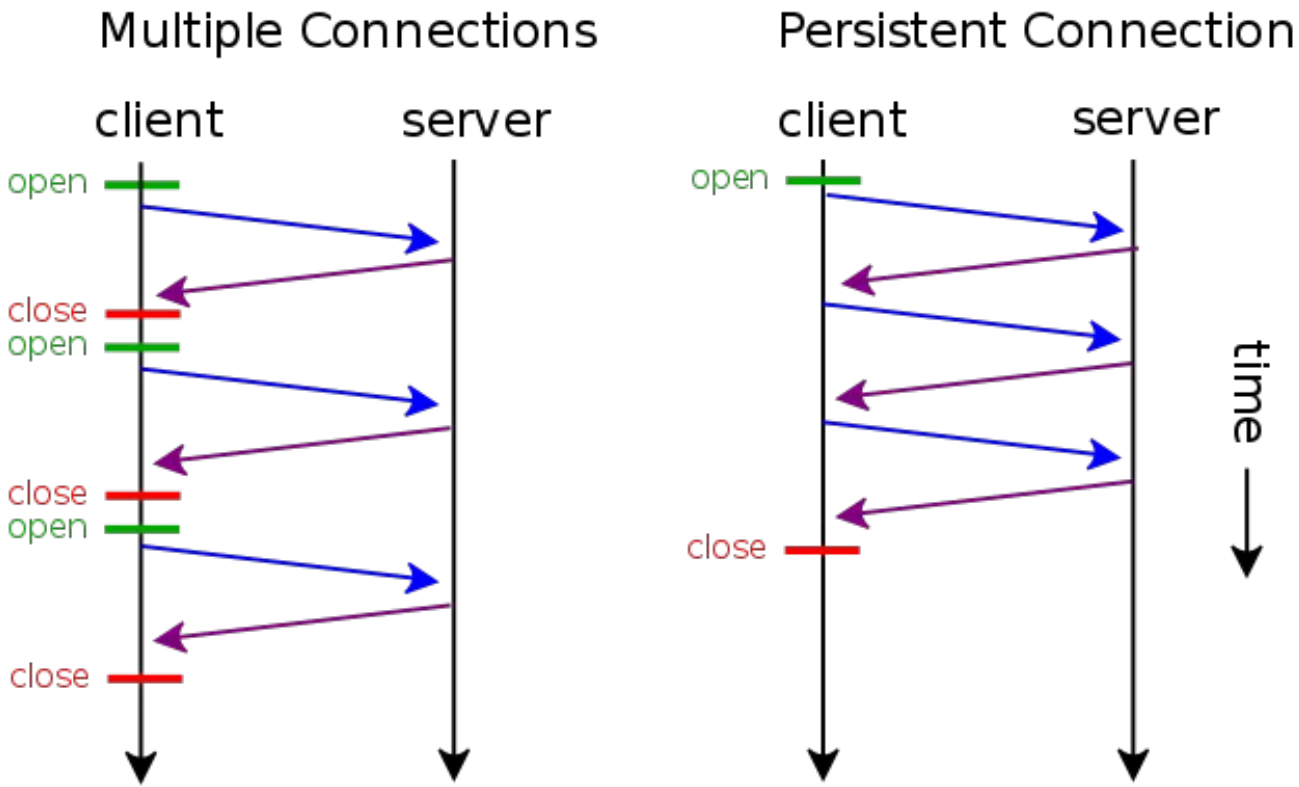
```go
func doSomeRequest() error {
    resp, err := http.DefaultClient.Get("http://example.com")
    if err != nil {
        return err
    }
    defer resp.Body.Close()

    if _, err := io.Copy(ioutil.Discard, resp.Body); err != nil {
        return err
    }

    return nil
}
```

HTTP Keep-Alive

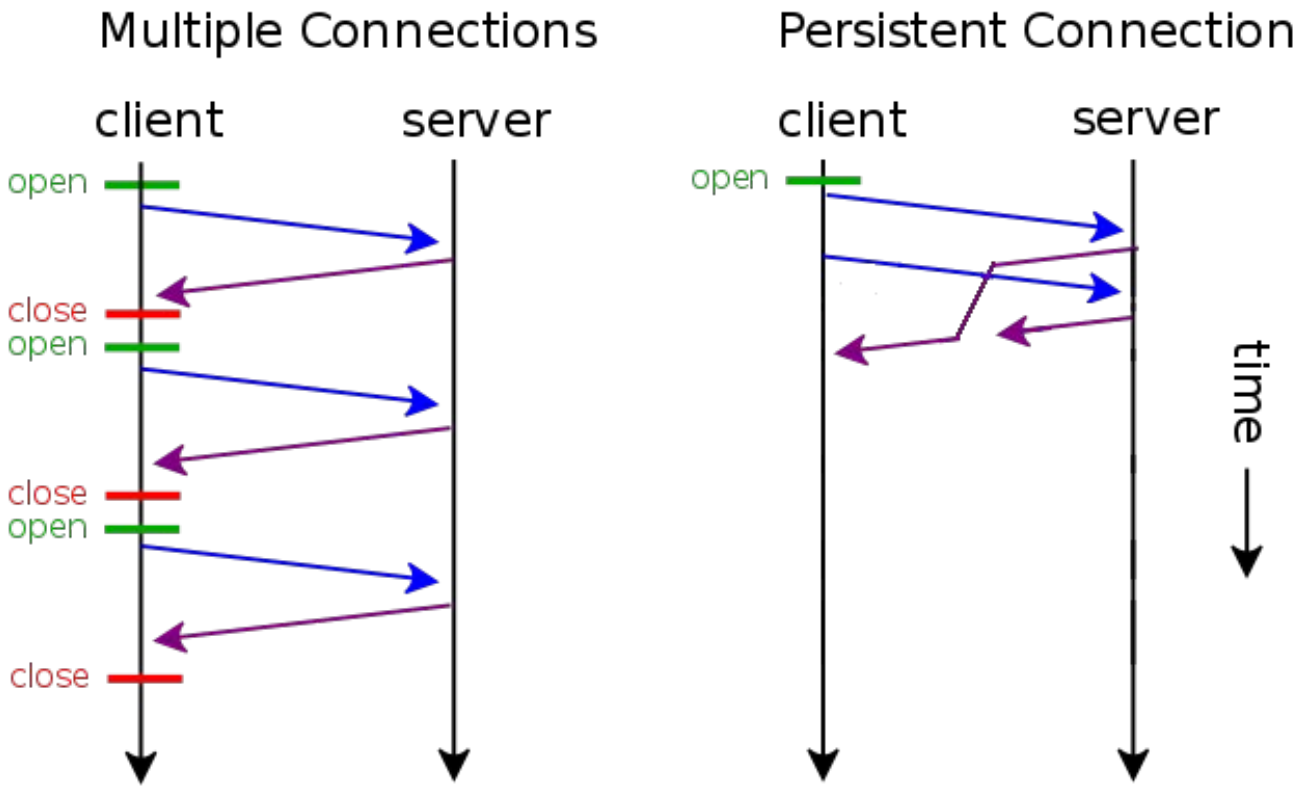
```
// Body represents the response body.
//
// The response body is streamed on demand as the Body field
// is read. If the network connection fails or the server
// terminates the response, Body.Read calls return an error.
//
// The http Client and Transport guarantee that Body is always
// non-nil, even on responses without a body or responses with
// a zero-length body. It is the caller's responsibility to
// close Body. The default HTTP client's Transport may not
// reuse HTTP/1.x "keep-alive" TCP connections if the Body is
// not read to completion and closed.
//
// The Body is automatically dechunked if the server replied
// with a "chunked" Transfer-Encoding.
//
// As of Go 1.12, the Body will also implement io.Writer
// on a successful "101 Switching Protocols" response,
// as used by WebSockets and HTTP/2's "h2c" mode.
Body io.ReadCloser
```



HTTP Keep-Alive

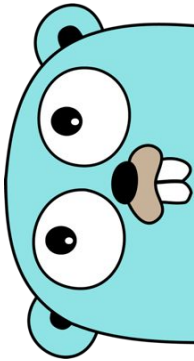


HTTP Keep-Alive



Why?

Why doesn’t Body.Close() do this for me?



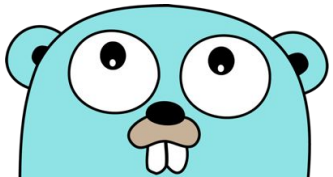
Why?

Why doesn’t Body.Close() do this for me?

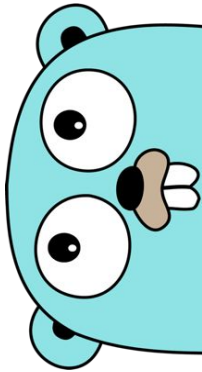
- 1GB bodies?
- Give the developer control

Server?

```
func SomeHandler(w http.ResponseWriter, r *http.Request) {

    // Is this needed?
    if _, err := io.Copy(ioutil.Discard, r.Body); err != nil {
        panic(err)
    }

    // ...

```



No



No, but actually….. Maybe!

```
// maxPostHandlerReadBytes is the max number of Request.Body bytes not
// consumed by a handler that the server will read from the client
// in order to keep a connection alive. If there are more bytes than
// this then the server to be paranoid instead sends a "Connection:
// close" response.
//
// This number is approximately what a typical machine's TCP buffer
// size is anyway.  (if we have the bytes on the machine, we might as
// well read them)
const maxPostHandlerReadBytes = 256 << 10
```

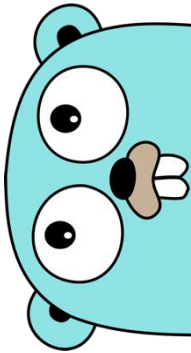
HTTP2

Is this still relevant with http2?

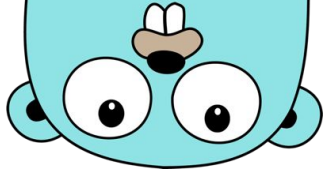
HTTP2

Is this still relevant with http2?

- No
  HTTP2 uses multiple streams per connection.



HTTP2



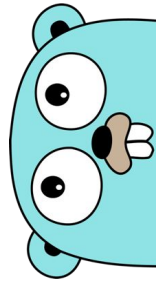
Is this still relevant with http2?

- No
- But actually Yes!, otherwise you'll get:

  `"http2: stream closed"`

  errors in your `net/http.Server.ErrorLog`

```go
func doNotDoThis() (int, error) {
    // DON'T DO THIS!!!
    // By not reusing this client you aren't using keep-alive!.
    client := http.Client{
        Transport: &http.Transport{
            MaxIdleConns: 10,
        },
    }

    resp, err := client.Get("http://example.com")
    if err != nil {
        return 0, err
    }
    defer resp.Body.Close()

    if _, err := io.Copy(ioutil.Discard, resp.Body); err != nil {
        return 0, err
    }

    return resp.StatusCode, nil
}
```



What about Trailers?

What are Trailers?

What if there is no body?

Questions?

Why doesn’t http.Client discard 256kb?

What about HTTP3?

Erik Dubbelboer
github.com/erikdubbelboer