Calls

jwilson edited this page on 2 Jan 2016 · 6 revisions

The HTTP client's job is to accept your request and produce its response. This is simple in theory but it gets tricky(狡猾棘手) in practice.(http client接收request并且response,虽然这句话很简单但是在实践中却很棘手)

Requests

Each HTTP request contains a URL, a method (like `GET` or `POST`), and a list of headers. Requests may also contain a body: a data stream of a specific content type.

Responses

The response answers the request with a code (like 200 for success or 404 for not found), headers, and its own optional(可选择的) body.

Rewriting Requests 重写请求

When you provide OkHttp with an HTTP request, you're describing(描写形容) the request at a high-level: *"fetch(去拿) me this URL with these headers."* For correctness(正确的) and efficiency(效率性), OkHttp rewrites your request before transmitting it.

OkHttp may add headers that are absent(缺席) from the original request, including `Content-Length`, `Transfer-Encoding`, `User-Agent`, `Host`, `Connection`, and `Content-Type`. It will add an `Accept-Encoding` header for transparent(清澈的) response compression unless the header is already present(如果没有header的话可以添加accept-encoding). If you've got cookies, OkHttp will add a `Cookie` header with them.(添加cookies)

Some requests will have a cached(贮藏) response. When this cached response isn't fresh, OkHttp can do a *conditional(条件的) GET* to download an updated response if it's newer than what's cached. This

▸ Pages 13

Using OkHttp

- Calls
- Connections
- Recipes
- Interceptors
- HTTPS
- TLS Configuration History
- Events
- 4.x API
  - okhttp
  - dnsoverhttps
  - logging-interceptor
  - sse
  - tls
  - urlconnection
  - mockwebserver
- 3.x API
  - okhttp
  - dnsoverhttps
  - logging-interceptor
  - sse
  - tls
  - urlconnection
  - mockwebserver

requires headers like `If-Modified-Since` and `If-None-Match` to be added.(可以跟新本地的缓存的请求响应体)

Rewriting Responses

If transparent compression was used, OkHttp will drop the corresponding(相当的) response headers `Content-Encoding` and `Content-Length` because they don't apply to the decompressed response body.(压缩是会丢掉header信息的)

If a conditional GET was successful, responses from the network and cache are merged as directed(规则,指导) by the spec.

Follow-up Requests

When your requested URL has moved, the webserver will return a response code like `302` to indicate(表明) the document's new URL. OkHttp will follow the redirect to retrieve(恢复) a final response.

If the response issues an authorization(授权) challenge, OkHttp will ask the `Authenticator(证明)` (if one is configured) to satisfy(使...满意) the challenge. If the authenticator supplies a credential, the request is retried with that credential included.(vCode的的授权的类似的做法)

Retrying Requests

Sometimes connections fail: either a pooled connection was stale and disconnected, or the webserver itself couldn't be reached. OkHttp will retry the request with a different route if one is available.(路由重试连接池重用)

Calls

With rewrites, redirects, follow-ups and retries, your simple request may yield many requests and responses. OkHttp uses `Call` to model the task of satisfying your request through however many intermediate requests and responses are necessary. Typically this isn't many! But it's comforting to know that your code will continue to work if your URLs are redirected or if you failover to an alternate IP address.(通过重写、重定向、跟踪和重试，您的简单请求可能会产生许多请求和响应.OkHttp使用调用来建模满足您的请求的任务，不管需要多少中间请求和响应。通常这不是很多！但是，知道如果您的URL被重定向或故障转移到另一个IP地址，您的代码将继续工作，这是一件令人欣慰的事情)

Calls are executed in one of two ways:

- **Synchronous:** your thread blocks until the response is readable.

- StackOverflow
- FAQs
- Works with OkHttp

Developing OkHttp

- Building
- Concurrency
- Contributing

Clone this wiki locally

- **Asynchronous:** you enqueue the request on any thread, and get called back on another thread when the response is readable.

Calls can be canceled from any thread. This will fail the call if it hasn't yet completed! Code that is writing the request body or reading the response body will suffer an `IOException` when its call is canceled.(可以在任何线程线程取消请求,若是已经完成将不能起作用,所以要在取消请求的的request和response中添加IOException)

Dispatch 派遣,分发

For synchronous calls, you bring your own thread and are responsible for managing how many simultaneous(同时发生) requests you make. Too many simultaneous connections wastes resources; too few harms latency(潜在因素). (同步请求需要自己管理启动了多少线程,同时发生了多少个,浪费资源,带来不利的因素)

For asynchronous calls, `Dispatcher` implements policy(策略) for maximum simultaneous requests. You can set maximums per-webserver (default is 5), and overall (default is 64).(设置per-webserver five thread,total 64)