Route

Whenever a network route is set up with page.route() or browser_context.route(), the `Route` object allows to handle the route.

Learn more about networking.

Methods

abort

Added in: v1.8

Aborts the route's request.

Usage

```
route.abort()
route.abort(**kwargs)
```

Arguments

- `error_code` str *(optional)*

  Optional error code. Defaults to `failed`, could be one of the following:

  - `'aborted'` - An operation was aborted (due to user action)
  - `'accessdenied'` - Permission to access a resource, other than the network, was denied
  - `'addressunreachable'` - The IP address is unreachable. This usually means that there is no route to the specified host or network.
  - `'blockedbyclient'` - The client chose to block the request.
  - `'blockedbyresponse'` - The request failed because the response was delivered along with requirements which are not met ('X-Frame-Options' and 'Content-Security-Policy' ancestor checks, for instance).

- `'connectionaborted'` - A connection timed out as a result of not receiving an ACK for data sent.
- `'connectionclosed'` - A connection was closed (corresponding to a TCP FIN).
- `'connectionfailed'` - A connection attempt failed.
- `'connectionrefused'` - A connection attempt was refused.
- `'connectionreset'` - A connection was reset (corresponding to a TCP RST).
- `'internetdisconnected'` - The Internet connection has been lost.
- `'namenotresolved'` - The host name could not be resolved.
- `'timedout'` - An operation timed out.
- `'failed'` - A generic failure occurred.

continue_

Added in: v1.8

Continues route's request with optional overrides.

Usage

Sync **Async**

```
def handle(route, request):
    # override headers
    headers = {
        **request.headers,
        "foo": "foo-value", # set "foo" header
        "bar": None # remove "bar" header
    }
    route.continue_(headers=headers)

page.route("**/*", handle)
```

Arguments

- `headers` Dict[str, str] *(optional)*

  If set changes the request HTTP headers. Header values will be converted to a string.

- `method` str *(optional)*

If set changes the request method (e.g. GET or POST).

- `post_data` str|bytes|Serializable *(optional)*

  If set changes the post data of request.

- `url` str *(optional)*

  If set changes the request URL. New URL must have same protocol as original one.

Details

Note that any overrides such as `url` or `headers` only apply to the request being routed. If this request results in a redirect, overrides will not be applied to the new redirected request. If you want to propagate a header through redirects, use the combination of route.fetch() and route.fulfill() instead.

fallback

Added in: v1.23

When several routes match the given pattern, they run in the order opposite to their registration. That way the last registered route can always override all the previous ones. In the example below, request will be handled by the bottom-most handler first, then it'll fall back to the previous one and in the end will be aborted by the first registered route.

Usage

Sync Async

```
page.route("**/*", lambda route: route.abort())  # Runs last.
page.route("**/*", lambda route: route.fallback())  # Runs second.
page.route("**/*", lambda route: route.fallback())  # Runs first.
```

Registering multiple routes is useful when you want separate handlers to handle different kinds of requests, for example API calls vs page resources or GET requests vs POST requests as in the example below.

Sync Async

```python
# Handle GET requests.
def handle_get(route):
    if route.request.method != "GET":
        route.fallback()
        return
  # Handling GET only.
  # ...

# Handle POST requests.
def handle_post(route):
    if route.request.method != "POST":
        route.fallback()
        return
  # Handling POST only.
  # ...

page.route("**/*", handle_get)
page.route("**/*", handle_post)
```

One can also modify request while falling back to the subsequent handler, that way intermediate route handler can modify url, method, headers and postData of the request.

Sync **Async**

```python
def handle(route, request):
    # override headers
    headers = {
        **request.headers,
        "foo": "foo-value", # set "foo" header
        "bar": None # remove "bar" header
    }
    route.fallback(headers=headers)

page.route("**/*", handle)
```

Arguments

- `headers` Dict[str, str] *(optional)*

  If set changes the request HTTP headers. Header values will be converted to a string.

- `method` str *(optional)*

If set changes the request method (e.g. GET or POST).

- `post_data` str|bytes|Serializable *(optional)*

  If set changes the post data of request.

- `url` str *(optional)*

  If set changes the request URL. New URL must have same protocol as original one. Changing the URL won't affect the route matching, all the routes are matched using the original request URL.

fetch

Added in: v1.29

Performs the request and fetches result without fulfilling it, so that the response could be modified and then fulfilled.

Usage

Sync **Async**

```
def handle(route):
    response = route.fetch()
    json = response.json()
    json["message"]["big_red_dog"] = []
    route.fulfill(response=response, json=json)

page.route("https://dog.ceo/api/breeds/list/all", handle)
```

Arguments

- `headers` Dict[str, str] *(optional)*

  If set changes the request HTTP headers. Header values will be converted to a string.

- `max_redirects` int *(optional)* Added in: v1.31

  Maximum number of request redirects that will be followed automatically. An error will be thrown if the number is exceeded. Defaults to `20`. Pass `0` to not follow redirects.

- `method` str *(optional)*

  If set changes the request method (e.g. GET or POST).

- `post_data` str|bytes|Serializable *(optional)*

  Allows to set post data of the request. If the data parameter is an object, it will be serialized to json string and `content-type` header will be set to `application/json` if not explicitly set. Otherwise the `content-type` header will be set to `application/octet-stream` if not explicitly set.

- `timeout` float *(optional)* Added in: v1.33

  Request timeout in milliseconds. Defaults to `30000` (30 seconds). Pass `0` to disable timeout.

- `url` str *(optional)*

  If set changes the request URL. New URL must have same protocol as original one.

Returns

- APIResponse

Details

Note that `headers` option will apply to the fetched request as well as any redirects initiated by it. If you want to only apply `headers` to the original request, but not to redirects, look into route.continue_() instead.

fulfill

Added in: v1.8

Fulfills route's request with given response.

Usage

An example of fulfilling all requests with 404 responses:

Sync **Async**

```
page.route("**/*", lambda route: route.fulfill(
    status=404,
    content_type="text/plain",
    body="not found!"))
```

An example of serving static file:

Sync **Async**

```
page.route("**/xhr_endpoint", lambda route: route.fulfill(path="mock_data.json"))
```

Arguments

- `body` str|bytes *(optional)*

  Response body.

- `content_type` str *(optional)*

  If set, equals to setting `Content-Type` response header.

- `headers` Dict[str, str] *(optional)*

  Response headers. Header values will be converted to a string.

- `json` Serializable *(optional)* Added in: v1.29

  JSON response. This method will set the content type to `application/json` if not set.

- `path` Union[str, pathlib.Path] *(optional)*

  File path to respond with. The content type will be inferred from file extension. If `path` is a relative path, then it is resolved relative to the current working directory.

- `response` APIResponse *(optional)* Added in: v1.15

  APIResponse to fulfill route's request with. Individual fields of the response (such as headers) can be overridden using fulfill options.

- `status` int *(optional)*

Response status code, defaults to `200`.

Properties

request

Added in: v1.8

A request to be routed.

Usage

```
route.request
```

Returns

- Request