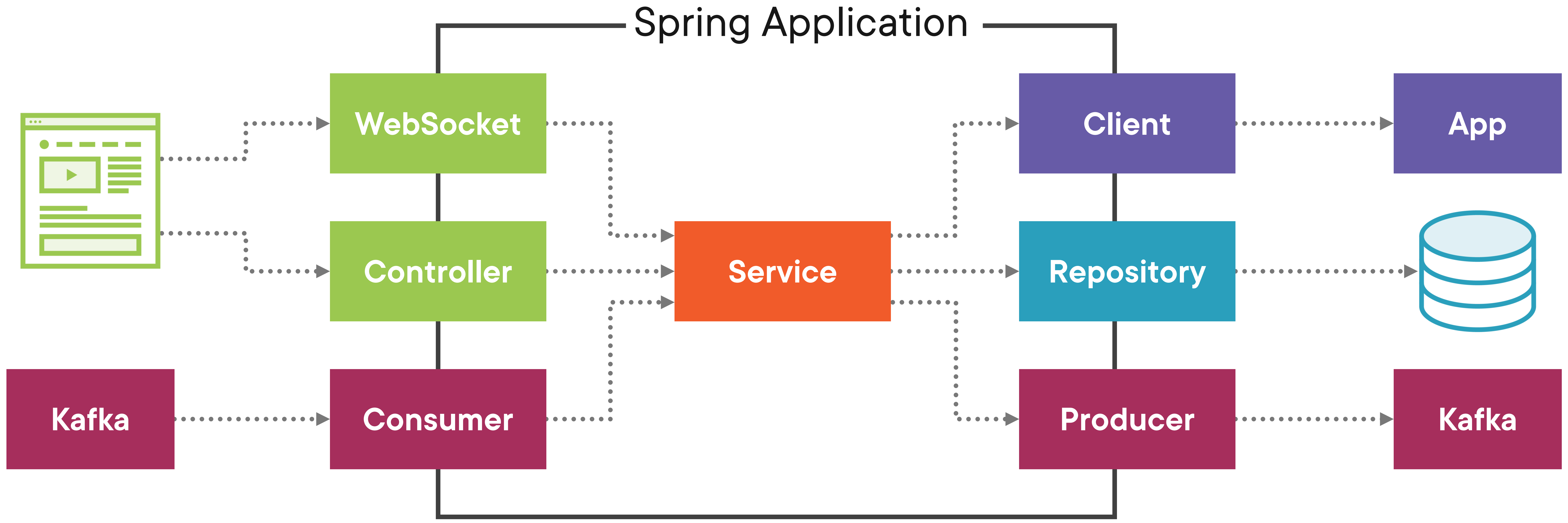
Calling REST APIs with WebClient

Bogdan Sucaciu
Tech Lead

@bsucaciu bsucaciu.com

Layered Architecture



WebClient

Reactor Netty

Jetty Reactive HttpClient

Apache HttpComponents

```
WebClient.create()

WebClient.create(String baseUrl)



WebClient.builder()
         .uriBuilderFactory(...)
         .defaultUriVariables(...)
         .defaultHeader(...)
         .defaultCookie(...)
         .defaultRequest(...)
         .filter(...)
         .exchangeStrategies(...)
         .clientConnector(...)
         .build()
```

◀ **Create WebClient**

◀ **Create WebClient with base URL**

◀ **Builder**

◀ **Provide base URL**

◀ **Default values to use when expanding URI**

◀ **Provide default headers**

◀ **Provide default cookies**

◀ **Customize request**

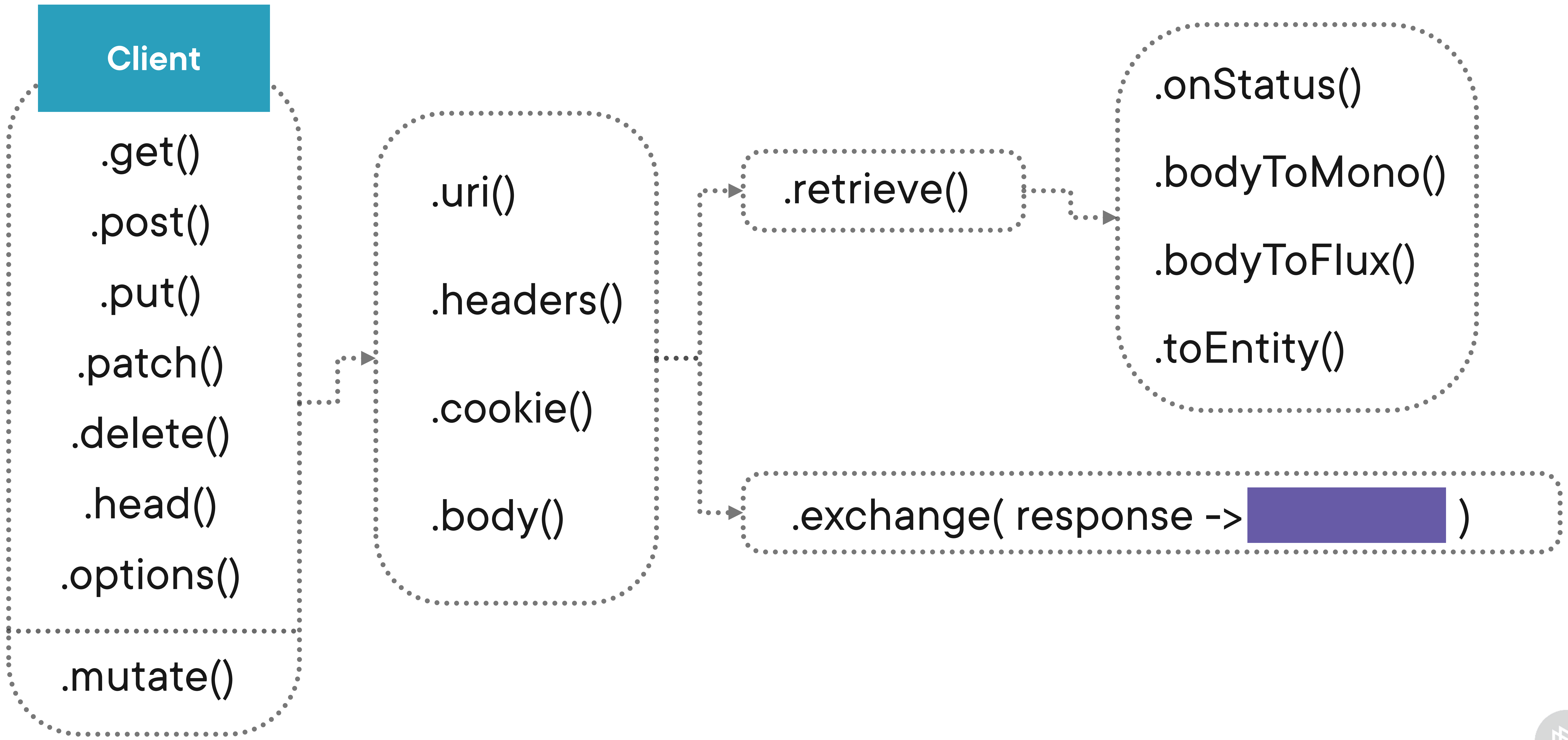
◀ **Client filter**

◀ **HTTP reader/writer customization**

◀ **HTTP client library settings**

◀ **Finalize build**

WebClient


Client
.get()
.post()
.put()
.patch()
.delete()
.head()
.options()
.mutate()
.uri()
.headers()
.cookie()
.body()
.retrieve()
.onStatus()
.bodyToMono()
.bodyToFlux()
.toEntity()
.exchange(response ->)

Demo

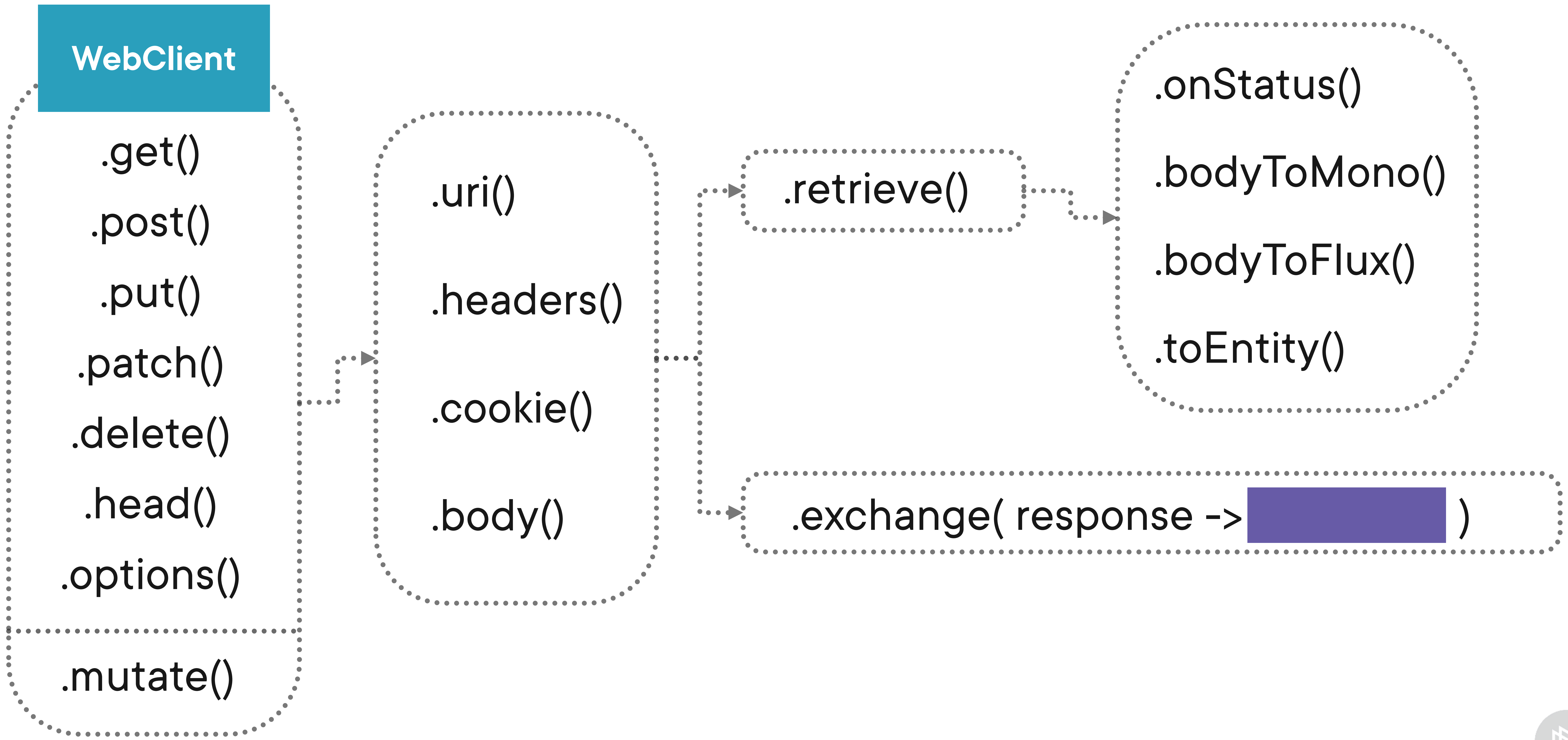
Build the Stock Market WebClient

Use WebClient to make GET calls

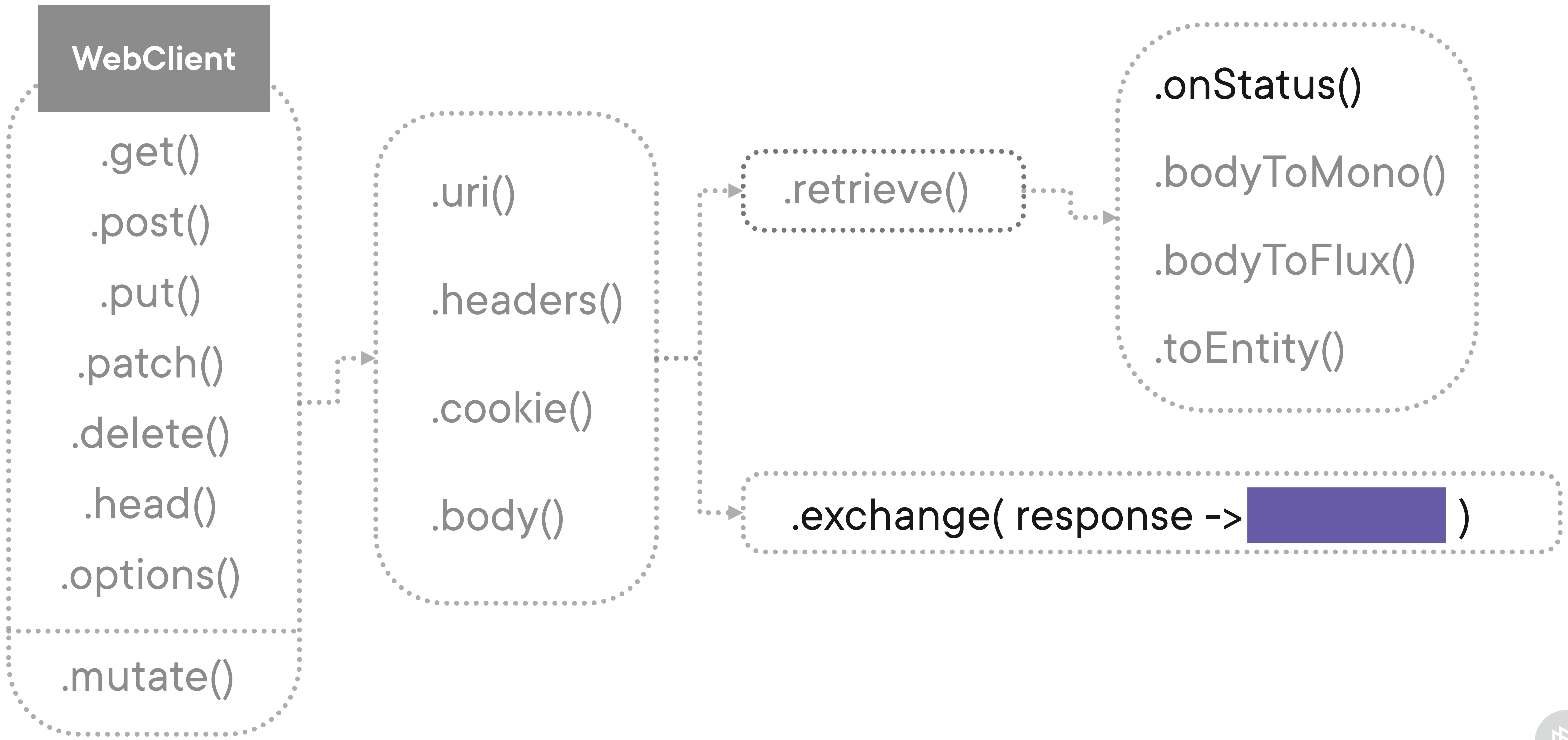
Use WebClient to make POST calls

Error Handling

WebClient


WebClient
.get()
.post()
.put()
.patch()
.delete()
.head()
.options()
.mutate()
.uri()
.headers()
.cookie()
.body()
.retrieve()
.onStatus()
.bodyToMono()
.bodyToFlux()
.toEntity()
.exchange(response ->)

WebClient


WebClient
.get()
.post()
.put()
.patch()
.delete()
.head()
.options()
.mutate()
.uri()
.headers()
.cookie()
.body()
.retrieve()
.onStatus()
.bodyToMono()
.bodyToFlux()
.toEntity()
.exchange(response ->)

Error Handling with OnStatus

```
webClient.post()
          .retrieve()
```

2xxResponse	`.bodyToMono(2xxResponse.class)`
4xxResponse	`.onStatus(HttpStatus::is4xxClientError, response -> 🟧 )`
5xxResponse	`.onStatus(HttpStatus::is5xxServerError, response -> 🟪 )`

Error Handling with OnStatus

```
webClient.post()
        .retrieve()
```

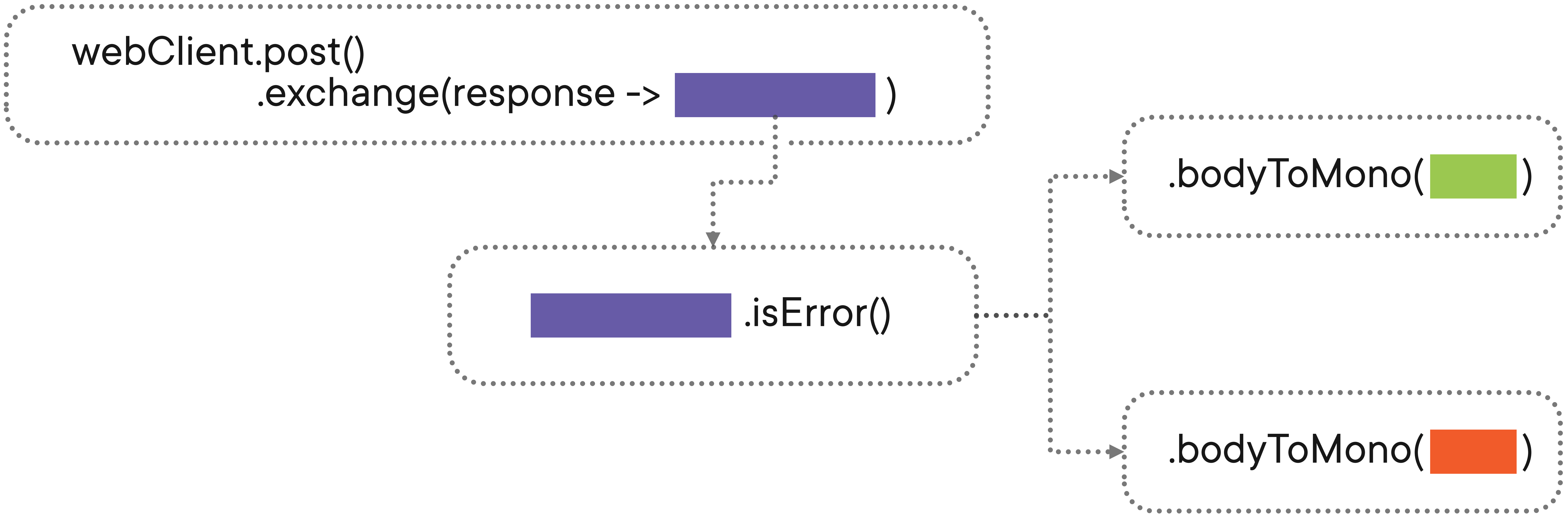
4xxResponse	`.onStatus(HttpStatus::is4xxClientError, response -> 🟧 )`
5xxResponse	`.onStatus(HttpStatus::is5xxServerError, response -> 🟪 )`
2xxResponse	`.bodyToMono(2xxResponse.class)`

Error Handling with OnStatus

```
response.createException()
            .flatMap(webClientResponseException -> Mono.error(  ))
```

```
response.bodyToMono(ErrorResponse.class)
            .flatMap(errorResponse -> Mono.error(  ))
```

Error Handling with Exchange

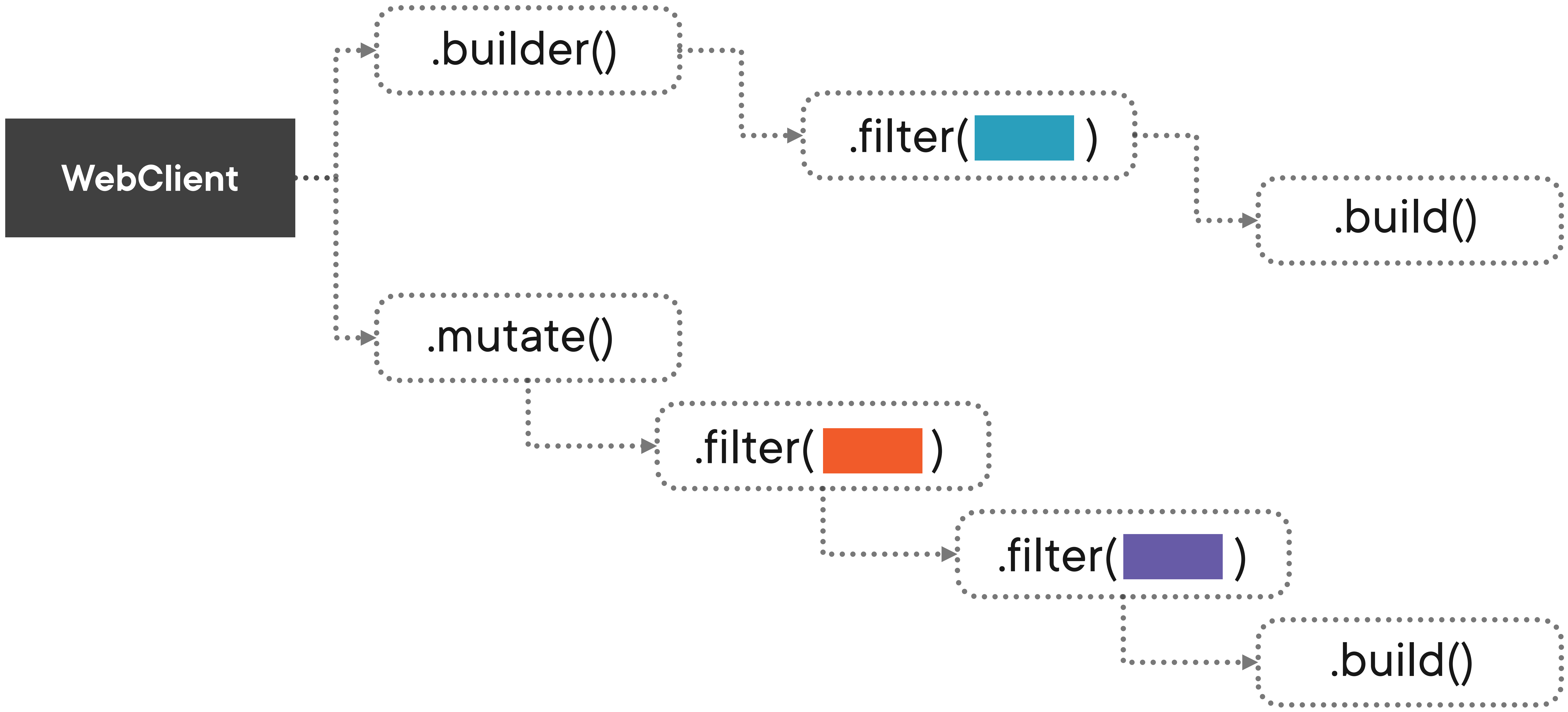


Demo

Error handling with exchange

Filters

WebClient Filters



WebClient Filters

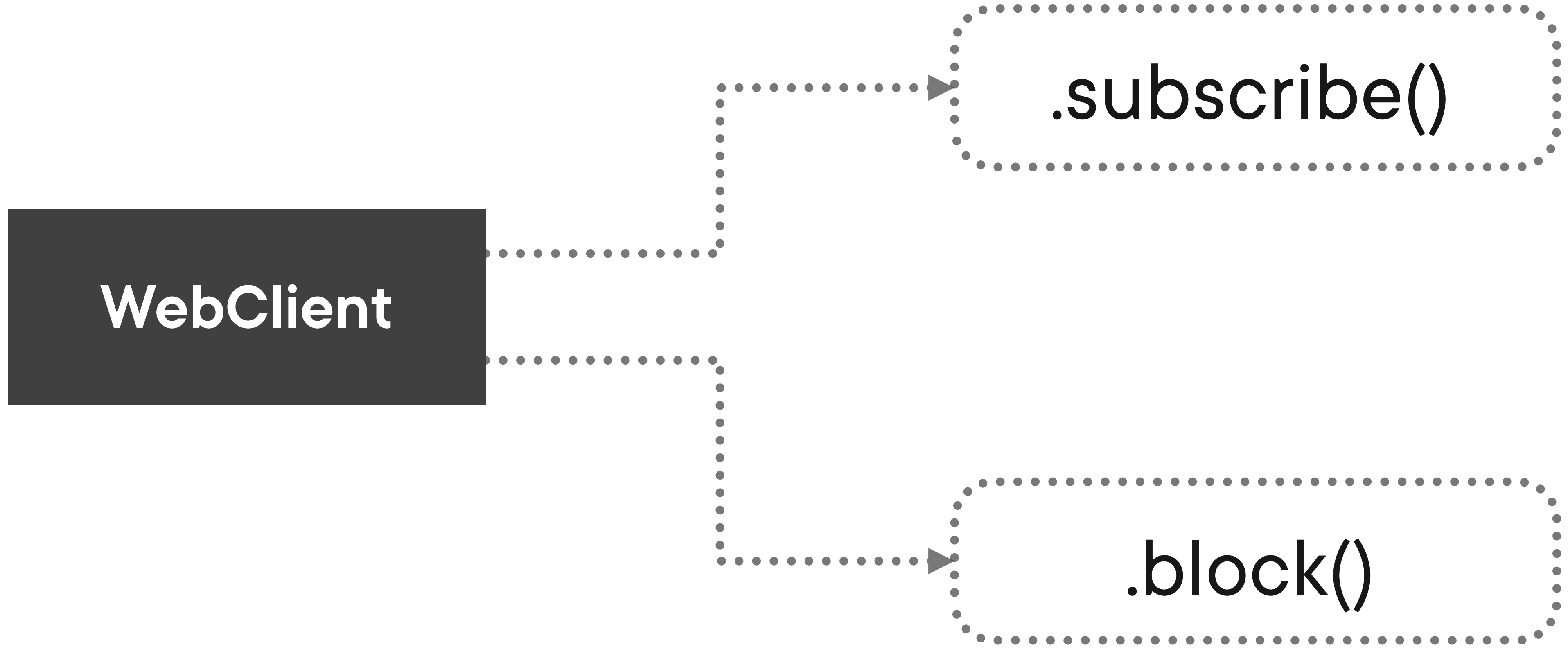
Lambda expression

(request, next) -> ...

Built-in exchange filter functions

Custom exchange filter functions

Synchronous Calls



Demo

Create custom filter

Up Next:
Testing Reactive Streams