目录

- 1. 请求头过滤器
- 2. 请求内容过滤器
- 3. 请求参数处理器
- 4. 请求数据块过滤器
- 5. 响应头过滤器
- 6. 响应内容过滤器
- 7. 响应参数处理器
- 8. 响应数据块过滤器

展开

1. 请求头过滤器

名称：requestHeadersFilter
说明：该过滤器可以在请求转发前，对请求头进行预处理。
入参：

参数名	说明
headers	即将进行转发的请求头，可以通过修改此对象上的值，改变实际转发出去的请求头内容
req	客户端请求对象，此对象挂载了一些常用的属性，如：method, parsedUrl, sp_options, socket 等
rule	处理当前请求的规则对象
rf	进程内共享的上下文引用对象，常用的属性有：writeLog, require, getConfig, redis, cache 等

出参：
该过滤器没有要求必须带返回内容，但如果过滤器内部需要做异步操作，可以通过返回一个 Promise 来表示这是一个异常操作。
如果需要提示中断当前处理流程，可通过返回一个特定的 Promise.reject：

```
return Promise.reject({ statusCode: 403, headers: {}, body: 'Access Denied!' });
```

或 throw 一个特定的 error 来达到效果：

```
throw { statusCode: 403, headers: {}, body: 'Access Denied!'  };
```

2. 请求内容过滤器

名称：requestBodyFilter
说明：该过滤器可以在请求包体转发前，对请求包体进行预处理。
入参：

参数名	说明
req	客户端请求对象，此对象挂载了一些常用的属性，如：method, parsedUrl, sp_options, socket 等
rule	处理当前请求的规则对象
rf	进程内共享的上下文引用对象，常用的属性有：writeLog, require, getConfig, redis, cache 等
body	使用 rule.reqBodyEncoding (默认 utf-8) 编码后请求包体内容，string 类型
buf	原始的请求包体内容，Buffer 类型

出参：
该过滤器明确要求必须返回处理后的 body 内容，可以返回 string 类型或 Buffer 类型。
但如果过滤器内部需要做异步操作，可以通过返回一个 Promise 来表示这是一个异常操作。
如果需要提示中断当前处理流程，可通过返回一个特定的 Promise.reject：

```
return Promise.reject({ statusCode: 403, headers: {}, body: 'Access Denied!' });
```

或 throw 一个特定的 error 来达到效果：

```

throw { statusCode: 403, headers: {}, body: 'Access Denied!'  };
```

3. 请求参数处理器

名称：requestParameterProcesser
说明：该处理器可以在请求包体转发前，对请求包体进行预处理，该处理器是“请求内容过滤器 ”的一个升级版本，通过 requestSourceContentType 或系统自动根据 Content-Type 识别，将纯文本的 body 转化为对象化的 param 参数。
入参：

参数名	说明
req	客户端请求对象，此对象挂载了一些常用的属性，如：method, parsedUrl, sp_options, socket 等
rule	处理当前请求的规则对象
rf	进程内共享的上下文引用对象，常用的属性有：writeLog, require, getConfig, redis, cache 等
param	对象化的请求参数，对象化过程是通过 requestSourceContentType 或系统自动根据 Content-Type 识别
body	使用 rule.reqBodyEncoding (默认 utf-8) 编码后请求包体内容，string 类型
buf	原始的请求包体内容，Buffer 类型

出参：
该处理器明确要求必须返回处理后的 param 内容，param 为对象类型。
但如果处理器内部需要做异步操作，可以通过返回一个 Promise 来表示这是一个异常操作。
如果需要提示中断当前处理流程，可通过返回一个特定的 Promise.reject：

```

    return Promise.reject({ statusCode: 403, headers: {}, body: 'Access Denied!' });
```

或 throw 一个特定的 error 来达到效果：

```

    throw { statusCode: 403, headers: {}, body: 'Access Denied!'  };
```

4. 请求数据块过滤器

名称：requestChunkFilter
说明：该过滤器可以在每一个请求数据块转发前，对数据块进行预处理。
入参：

参数名	说明
req	客户端请求对象，此对象挂载了一些常用的属性，如：method, parsedUrl, sp_options, socket 等
rule	处理当前请求的规则对象
rf	进程内共享的上下文引用对象，常用的属性有：writeLog, require, getConfig, redis, cache 等
data	请求数据块，Buffer 类型

出参：
该过滤器明确要求必须返回处理后的 data 内容，返回值可是 string 或 Buffer 类型。

5. 响应头过滤器

名称：responseHeadersFilter
说明：该过滤器可以在响应转发前，对响应头进行预处理。
入参：

参数名	说明
headers	即将进行转发的响应头，可以通过修改此对象上的值，改变实际转发出去的响应头内容
req	客户端请求对象，此对象挂载了一些常用的属性，如：method, parsedUrl, sp_options, socket 等
res	远程服务响应对象
rule	处理当前请求的规则对象
rf	进程内共享的上下文引用对象，常用的属性有：writeLog, require, getConfig, redis, cache 等

出参：
该过滤器没有要求必须带返回内容，但如果过滤器内部需要做异步操作，可以通过返回一个 Promise 来表示这是一个异常操作。
如果需要提示中断当前处理流程，可通过返回一个特定的 Promise.reject：

```

    return Promise.reject({ statusCode: 403, headers: {}, body: 'Access Denied!' });
```

或 throw 一个特定的 error 来达到效果：

```

    throw { statusCode: 403, headers: {}, body: 'Access Denied!'  };
```

6. 响应内容过滤器

名称：responseBodyFilter
说明：该过滤器可以在响应包体转发前，对响应包体进行预处理。
入参：

参数名	说明
req	客户端请求对象，此对象挂载了一些常用的属性，如：method, parsedUrl, sp_options, socket 等
res	远程服务响应对象
rule	处理当前请求的规则对象
rf	进程内共享的上下文引用对象，常用的属性有：writeLog, require, getConfig, redis, cache 等
body	使用 rule.reqBodyEncoding (默认 utf-8) 编码后请求包体内容，string 类型
buf	原始的请求包体内容，Buffer 类型

出参：
该过滤器明确要求必须返回处理后的 body 内容，可以返回 string 类型或 Buffer 类型。
但如果过滤器内部需要做异步操作，可以通过返回一个 Promise 来表示这是一个异常操作。
如果需要提示中断当前处理流程，可通过返回一个特定的 Promise.reject：

```

    return Promise.reject({ statusCode: 403, headers: {}, body: 'Access Denied!' });
```

或 throw 一个特定的 error 来达到效果：

```

    throw { statusCode: 403, headers: {}, body: 'Access Denied!'  };
```

7. 响应参数处理器

名称：responseParameterProcesser
说明：该处理器可以在响应包体转发前，对响应包体进行预处理，该处理器是“响应内容过滤器 ”的一个升级版本，通过 responseSourceContentType 或系统自动根据 Content-Type 识别，将纯文本的 body 转化为对象化的 param 参数。
入参：

参数名	说明
req	客户端请求对象，此对象挂载了一些常用的属性，如：method, parsedUrl, sp_options, socket 等
res	远程服务响应对象
rule	处理当前请求的规则对象
rf	进程内共享的上下文引用对象，常用的属性有：writeLog, require, getConfig, redis, cache 等
param	对象化的请求参数，对象化过程是通过 requestSourceContentType 或系统自动根据 Content-Type 识别
body	使用 rule.reqBodyEncoding (默认 utf-8) 编码后请求包体内容，string 类型
buf	原始的请求包体内容，Buffer 类型

出参：
该处理器明确要求必须返回处理后的 param 内容，param 为对象类型。
但如果处理器内部需要做异步操作，可以通过返回一个 Promise 来表示这是一个异常操作。
如果需要提示中断当前处理流程，可通过返回一个特定的 Promise.reject：

```

    return Promise.reject({ statusCode: 403, headers: {}, body: 'Access Denied!' });
```

或 throw 一个特定的 error 来达到效果：

```
1.
2.    throw { statusCode: 403, headers: {}, body: 'Access Denied!' };
```

8. 响应数据块过滤器

名称：responseChunkFilter
说明：该过滤器可以在每一个响应数据块转发前，对数据块进行预处理。
入参：

参数名	说明
req	客户端请求对象，此对象挂载了一些常用的属性，如：method, parsedUrl, sp_options, socket 等
res	远程服务响应对象
rule	处理当前请求的规则对象
rf	进程内共享的上下文引用对象，常用的属性有：writeLog, require, getConfig, redis, cache 等
data	请求数据块，Buffer 类型

出参：
该过滤器明确要求必须返回处理后的 data 内容，返回值可是 string 或 Buffer 类型。