

[AIWriteSupporter] Node Express server

- [@waylaidwanderer/fetch-event-source](#) Node Express server .

modules/FetchEventSource.js

```
const { fetch, Headers, Request, Response } = require('fetch-undici');

if (!globalThis.fetch) {
  globalThis.fetch = fetch;
  globalThis.Headers = Headers;
  globalThis.Request = Request;
  globalThis.Response = Response;
}

module.exports = require('@waylaidwanderer/fetch-event-source');
```

- GPT

server.js

```
const express = require('express');
const bodyParser = require('body-parser');
const { fetchEventSource } = require('./modules/FetchEventSource');

const GPT_API_URL = 'https://api.openai.com/v1/chat/completions'; // API URL
const GPT_API_KEY = ''; // API KEY

const app = express();
const router = express.Router();

router.post('/request', (request, response) => {
  const bodyData = Object.assign({}, request.body, {
    model: 'gpt-3.5-turbo',
    stream: true
  });

  requestGPT(response, bodyData)
    .then(() => response.end())
    .catch((error) => response.status(error.status).json(error).end());
});

/**
 * GPT .
 * @param {Response} response
 * @param {Object} bodyData
 * @returns {Promise}
 */
function requestGPT(response, bodyData) {
  const abortController = new AbortController();
  return new Promise(async (resolve, reject) => {
    try {
      await fetchEventSource(GPT_API_URL, {
        method: 'post',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${GPT_API_KEY}` // OpenAI
          // 'Api-Key': GPT_API_KEY` // Azure OpenAI
        },
        body: JSON.stringify(bodyData),
        signal: abortController.signal,
        onopen: async (openResponse) => {
          if (openResponse.status === 200) {
            response.on('close', () => {

```

```

        abortController.abort();
        resolve();
    });
    response.set({
      'Cache-Control': 'no-cache',
      'Connection': 'keep-alive',
      'Content-Type': 'text/event-stream'
    });
    response.flushHeaders();
    return;
  }
  let error;
  try {
    const json = await openResponse.json();
    error = json;
  } catch (e) {
    error = e;
  }
  error.status = openResponse.status;
  throw error;
},
onclose: () => resolve(),
onerror: (error) => {
  reject(error);
  throw error;
},
onmessage: (message) => {
  if (!message.data || message.event !== 'ping') {
    return;
  }
  response.write('data:' + message.data + '\n\n');
},
});
} catch (e) {
  reject(e);
}
});
}

app.use(bodyParser.json());
app.use('/', router);
app.listen(8080);

```

- X

server.js

```
const express = require('express');
const bodyParser = require('body-parser');

const HCX_API_URL = 'https://clovastudio.apigw.ntruss.com/testapp/v1/chat-completions/HCX-002'; // API URL
const HCX_API_KEY = ''; // API KEY
const HCX_GATEWAY_KEY = ''; // GATEWAY KEY

const app = express();
const router = express.Router();

router.post('/request', (request, response) => {
  const bodyData = request.body;

  requestHCX(response, bodyData)
    .then(() => response.end())
    .catch(err => {
      console.error('[ERROR] requestHCX chat completion failed: ' + err);
      return res.status(err.status || 500).json(err).end();
    });
});

/**
 * X .
 * @param {Response} response
 * @param {Object} body
 * @returns {Promise}
 */
function requestHCX(response, bodyData) {
  const source = axios.CancelToken.source();
  const url = HCX_API_URL;
  const headers = {
    'Content-Type': 'application/json',
    'Accept': 'text/event-stream',
    'X-NCP-CLOVASTUDIO-API-KEY': HCX_API_KEY,
    'X-NCP-APIGW-API-KEY': HCX_GATEWAY_KEY
  }
  return new Promise(async (resolve, reject) => {
    try {
      axios.post(url, bodyData, {
        headers,
        responseType: 'stream',
        cancelToken: source.token
      }).then(res => {
        res.data.pipe(res);
      }).catch(reject);

      response.on('close', () => {
        source.cancel('');
        resolve();
      });
    } catch (e) {
      reject(e);
    }
  });
}

app.use(bodyParser.json());
app.use('/', router);
app.listen(8080);
```