

# ソースオープンのススメ

---

v 2024 09  
石崎 充良

# 自己紹介

---

石崎 充良 ( @mishi\_cs )

C# Tokyo コミュニティ管理メンバー

**GitHub :**

<https://github.com/m-ishizaki>

**blog :**

<https://rksoftware.hatenablog.com/>



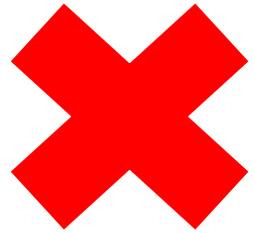
1. **(分量：03 ページ) ソースオープンとは**
2. (分量：05 ページ) 人はなぜブログを書くのか
3. (分量：02 ページ) とはいえそんな分量は書けない
4. (分量：04 ページ) 今日のために Pleasanter に AI を組み込んでみた
5. (分量：01 ページ) まとめ

---

目次

# ソースオープンとは

---



OSS とはオープンなソースのソフトウェア  
では **ありません**



Open Source Initiative (OSI) という組織が認め  
たライセンスを採用しているソースコードと解釈  
しておけばよいでしょう。

**\*\*要件にソースがオープン\*\*** というのがあります。

# OSS にしなくてよいのか？

OSS に**してください**

Open Source Initiative  
(OSI) という組織が認めた  
ライセンス(定番ライセンス)

誰でも利用可能

例えば OSS は軍事利用が可能です

軍事利用が可能ですないならば OSS ではありません

m-ishizaki / CSTokyoCSharpOdai Public  
forked from csharp-tokyo/CSharpOdai

<> Code Pull requests 1 Actions Projects Security Insights

main CSTokyoCSharpOdai / LICENSE

Permissions	Limitations	Conditions
✓ Commercial use	✗ Liability	① License and cop notice
✓ Modification	✗ Warranty	
✓ Distribution		

m-ishizaki/CSTokyoCSharpOdai is licensed under the MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and large derivatives may be distributed under different terms and without source code.

This is not legal advice. [Learn more about repository licenses](#)

m-ishizaki Initial commit

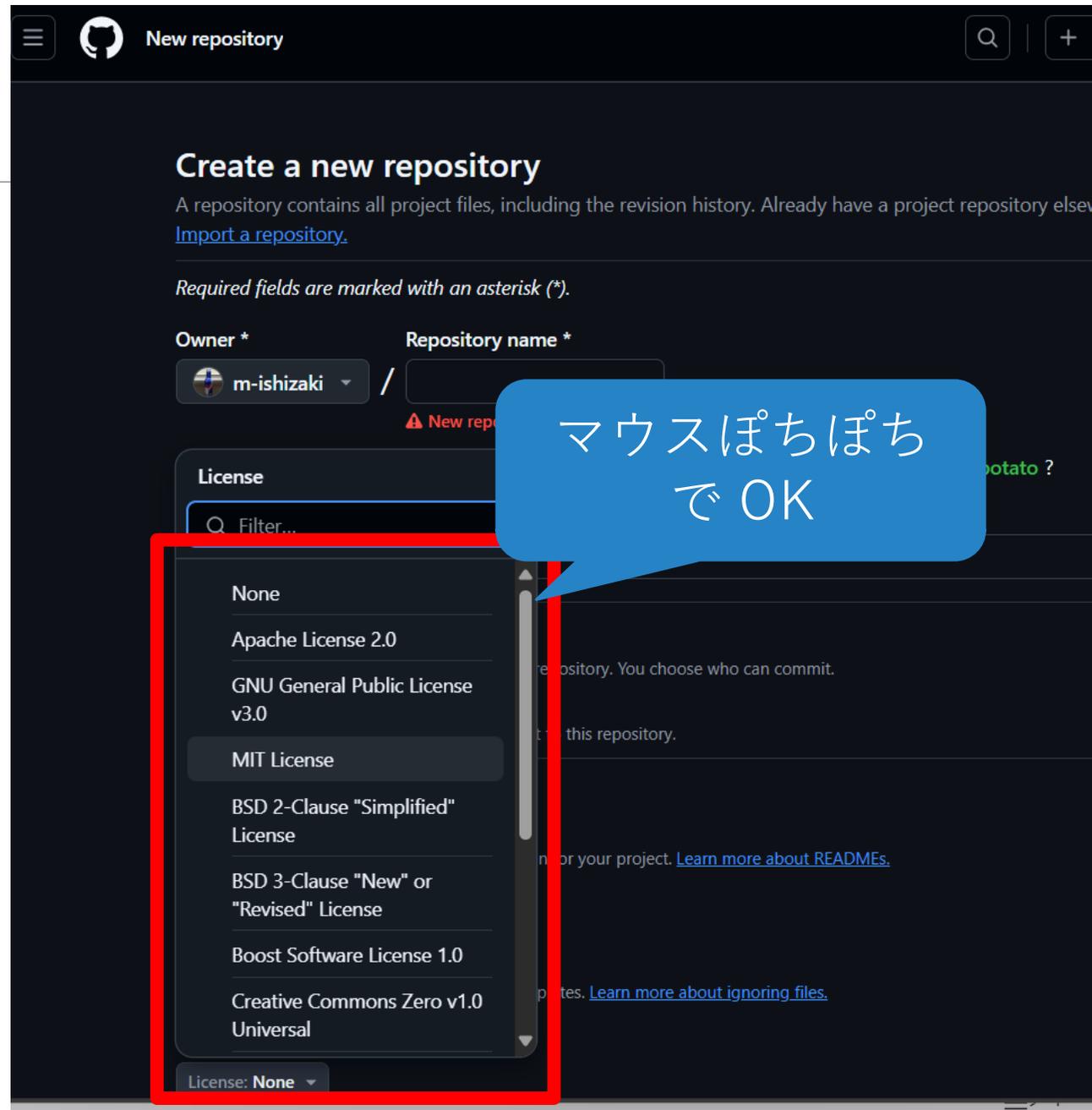
21 lines (17 loc) · 1.04 KB

```
1 MIT License
2
3 Copyright (c) 2024 C# Tokyo
4
5 Permission is hereby granted, free of charge, to any person obtaining a copy
6 of this software and associated documentation files (the "Software"), to deal
7 in the Software without restriction, including without limitation the rights
8 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9 copies of the Software, and to permit persons to whom the Software is
10 furnished to do so, subject to the following conditions:
11
12 The above copyright notice and this permission notice shall be included in all
13 copies or substantial portions of the Software.
14
15 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
20 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
21 SOFTWARE.
```

定番ライセンスは  
読まなくてよい

# 適用が簡単

GitHub でリポジトリを作るときに  
**マウスをカチカチするだけ**で適用できる、親切設計



1. (分量：03 ページ) ソースオープンとは
2. **(分量：05 ページ) 人はなぜブログを書くのか**
3. (分量：02 ページ) とはいえそんな分量は書けない
4. (分量：04 ページ) 今日のために Pleasanter に AI を組み込んでみた
5. (分量：01 ページ) まとめ

---

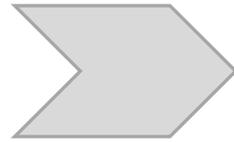
## 目次

# 人は忘れる生き物だから

---

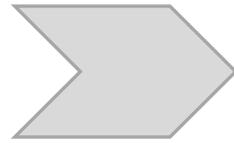
学んだことは書き留めておかねばなりません

どこに？



紙のノート？ 自宅 PC 上のファイル？  
クラウドストレージ？

どうやって  
読み返す？



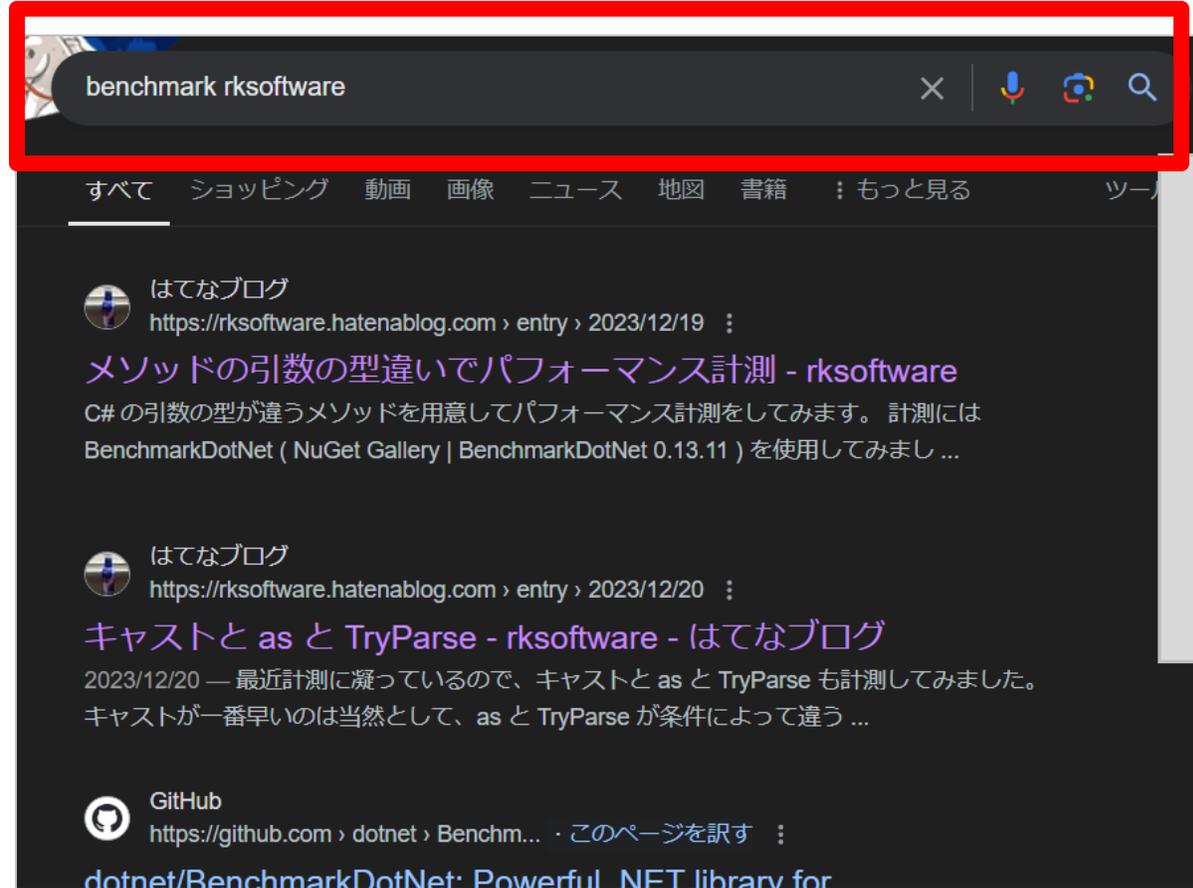
ノートを探してページを探して？ 自宅に帰って  
フォルダをたどって？



いつでも、どこでも Web 検索で  
読み返せたら最強なのは！

# 例えば

C# で早いコードを書きたいけど、ベンチマークの取り方どうやるんだっけ？



**rksoftware**  
Visual Studio とか C# とかが好きです

## メソッドの引数の型違いでパフォーマンス計測

C# の引数の型が違うメソッドを用意してパフォーマンス計測をしてみます。

計測には BenchmarkDotNet ( [NuGet Gallery](#) | [BenchmarkDotNet 0.13.11](#) ) を使用してみました。

### ■ 検証対象コード

引数違いで 3 つのメソッドを用意してみました。object 型のもの、int 型のもの、string 型のもので。

```
namespace ClassLibrary1
{
    public static class MethodExtensions
    {
        public static int ExtensionMethod(this object s) => 0;
        public static int ExtensionMethod(this int s) => 0;
        public static int ExtensionMethod(this string s) => 0;
    }
}
```

BenchmarkDotNet を使った計測コードです。

```
using ClassLibrary1;

BenchmarkDotNet.Running.BenchmarkRunner.Run<Sample>(new BenchmarkDotNet.Configs.DebugInProcessConfig());

public class Sample
{
    object o = new object();
    int i = 0;
    string s = "";

    [BenchmarkDotNet.Attributes.Benchmark]
    public void MethodObject() => o.ExtensionMethod();
}
```

# 不十分・わかりにくい

調べものをしていて、良さそうな記事を見つけました。喜んで読んでみると

内容が不十分

説明不足

わかりにくい

記事を書いた **本人にとって** 十分でわかりやすい

# 例えば

C# で早いコードを  
マークの取り方どう

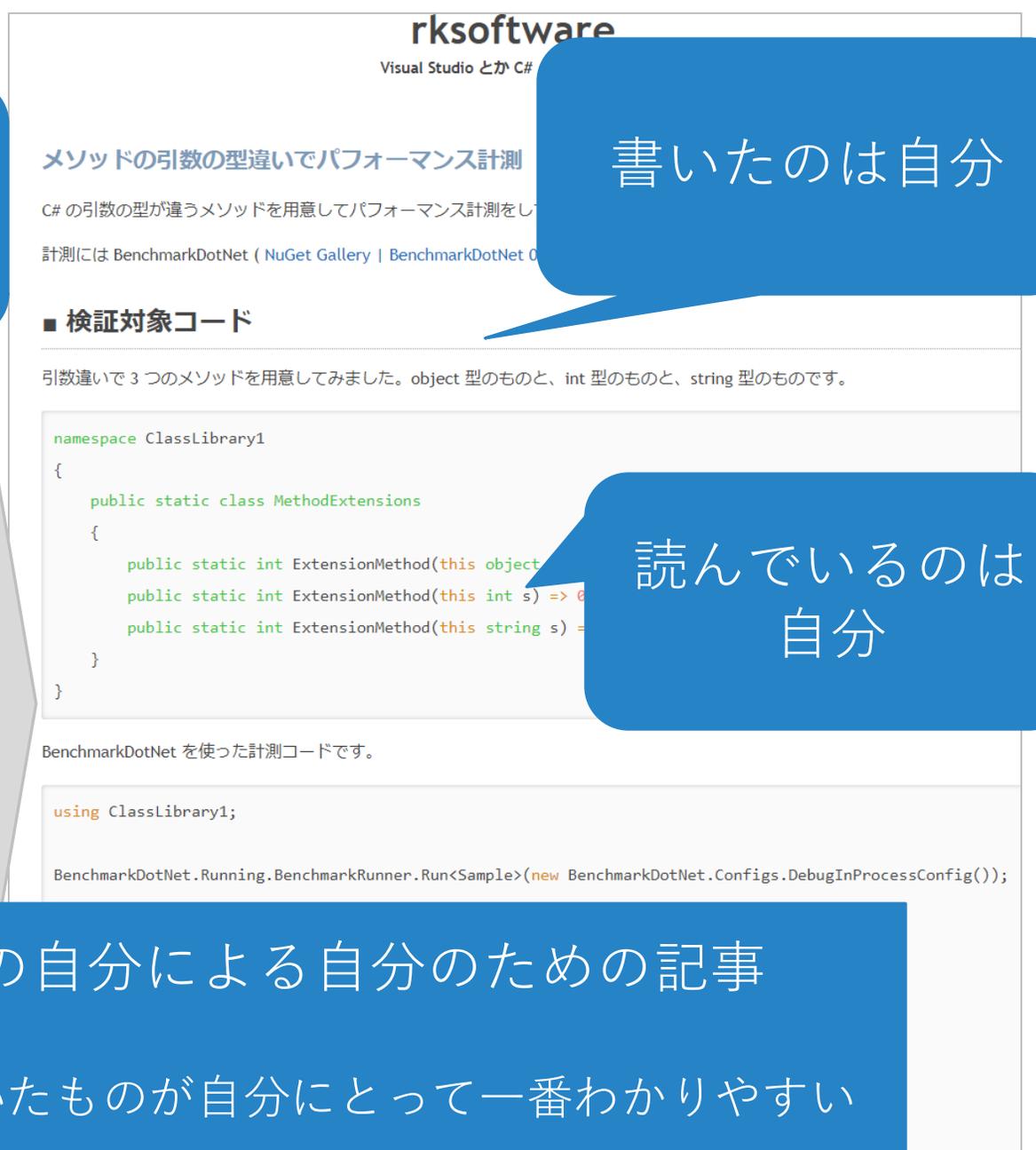
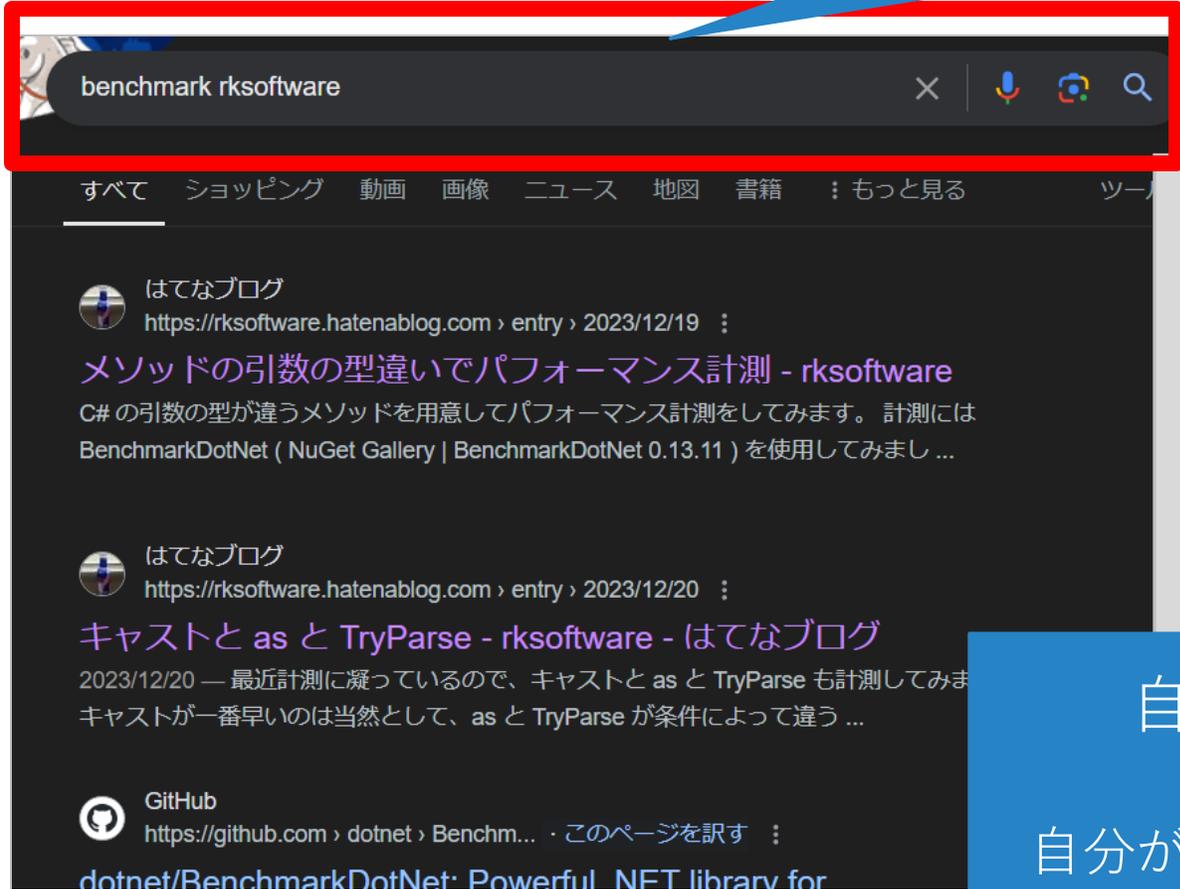
検索したのは自分

書いたのは自分

読んでいるのは  
自分

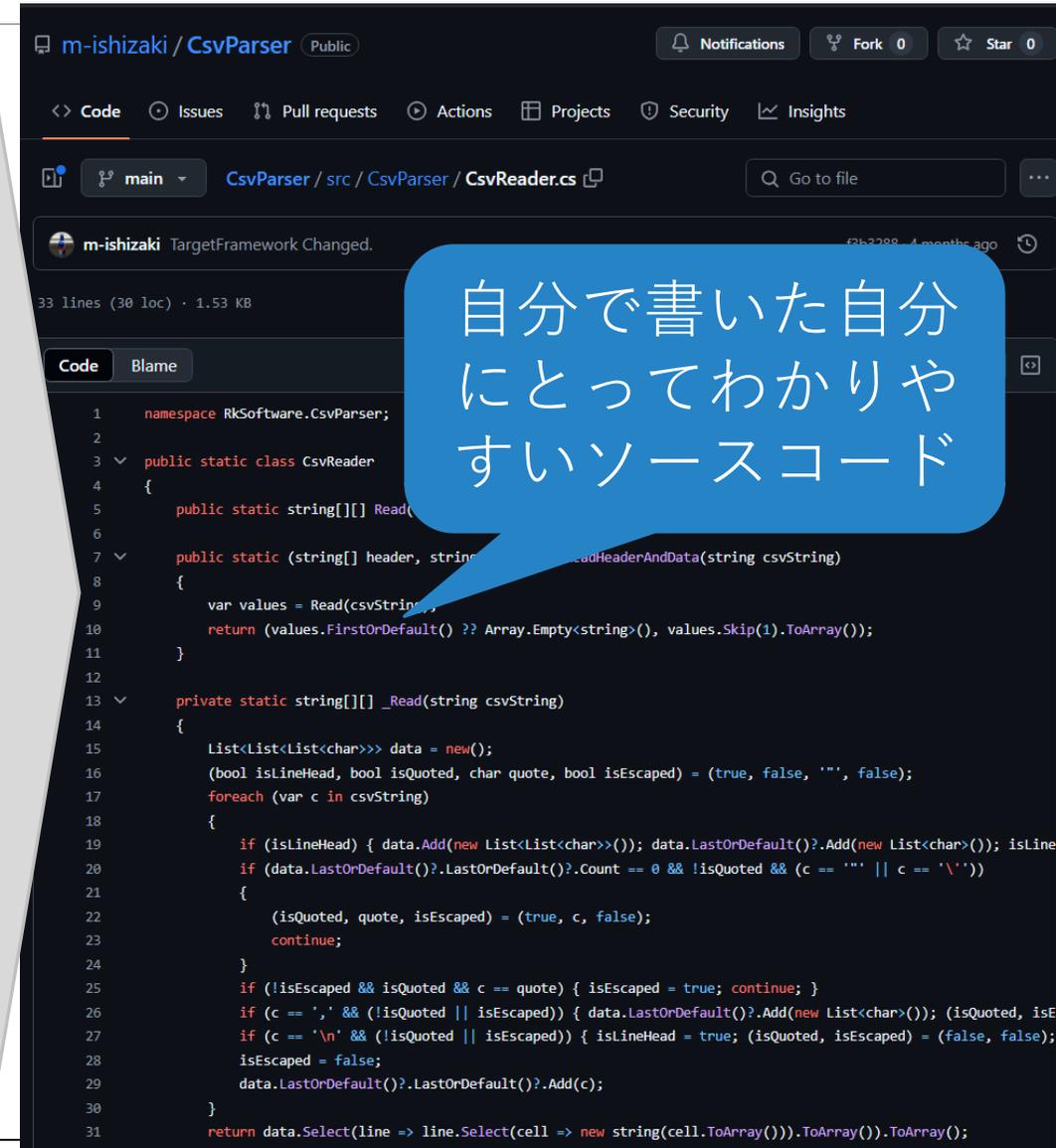
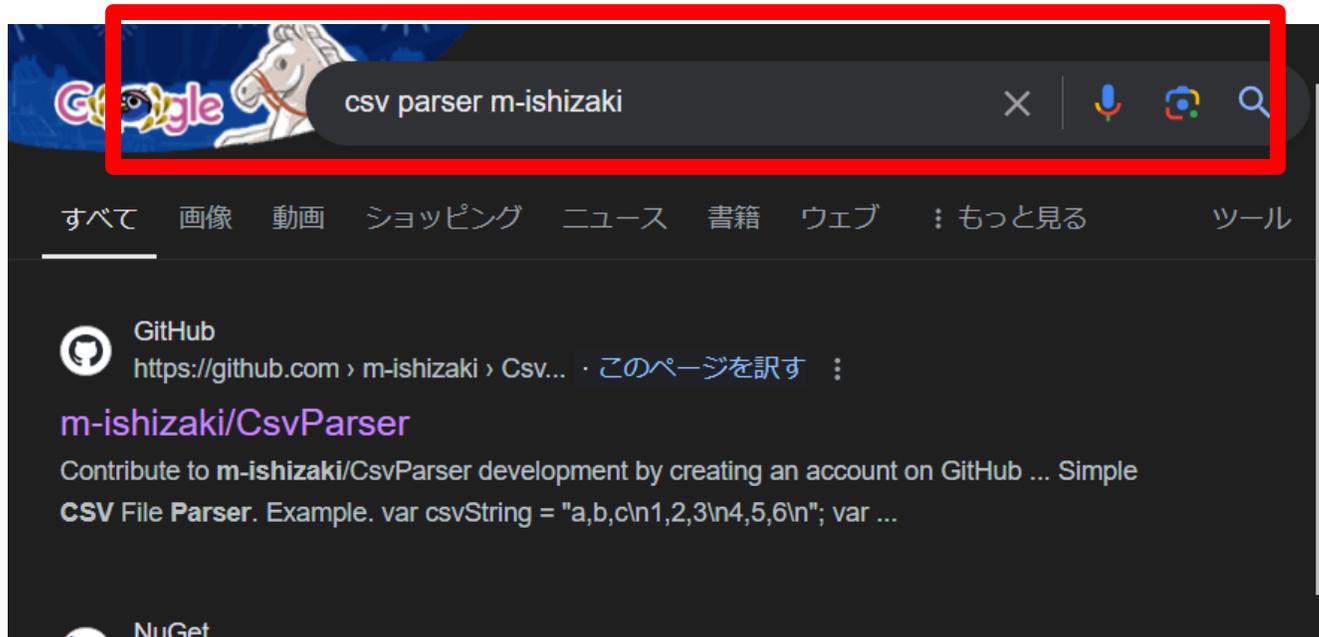
自分の自分による自分のための記事

自分が書いたものが自分にとって一番わかりやすい



# プログラム ソースコード も同じです

CSV を読み込むロジックってどうだったっけ？



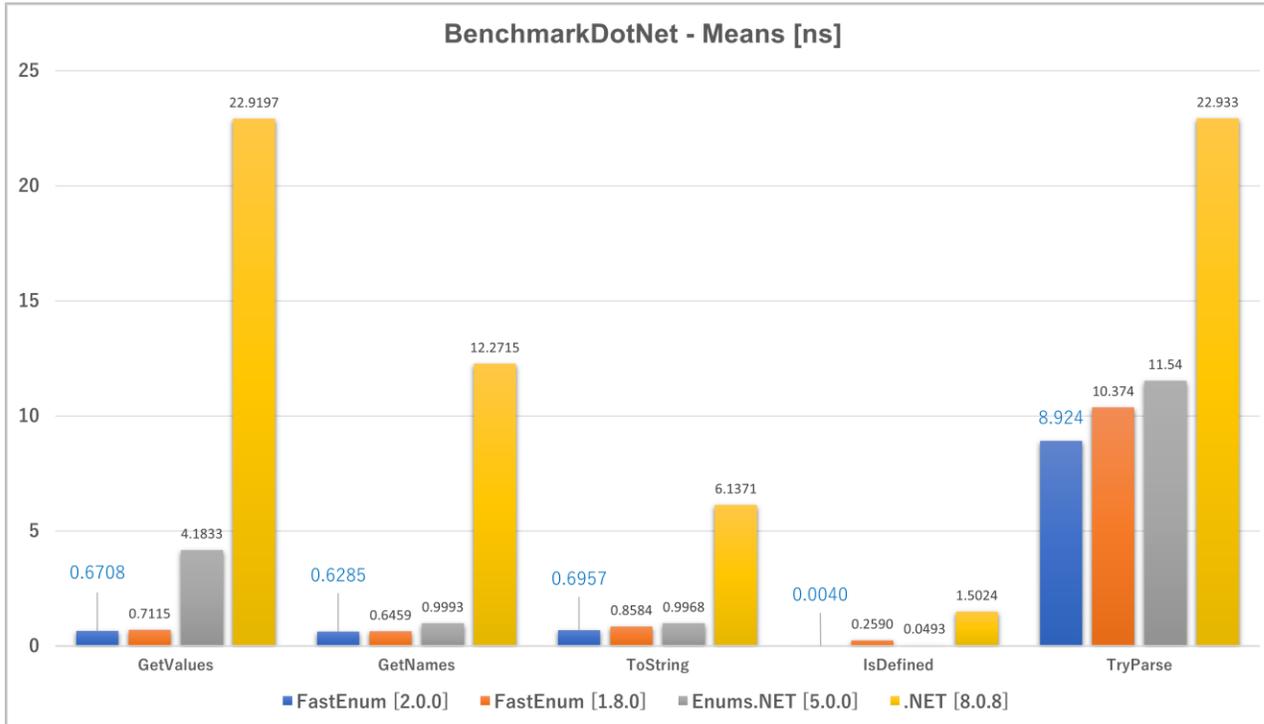
1. (分量：03 ページ) ソースオープンとは
2. (分量：05 ページ) 人はなぜブログを書くのか
3. **(分量：02 ページ) とはいえそんな分量は書けない**
4. (分量：04 ページ) 今日のために Pleasanter に AI を組み込んでみた
5. (分量：01 ページ) まとめ

---

## 目次

# 大丈夫です

## C# 界最速の enum ライブラリ



ダウンロードして  
ソースコード部分  
(8万文字程度のサイズ)

サイズ: 81.3 KB (83,277 バイト)  
ディスク上のサイズ: 100 KB (102,400 バイト)  
内容: ファイル数: 13、フォルダ数: 1

※ Pleasanter には 1 ファイル 5 百万文字 5MB あり

xin9le / FastEnum Public

Notifications Fork 24 Star 330

Code Issues Pull requests Actions Projects Security Insights

main Go to file Code About

xin9le Merge pull request ... fa9d4f3 · yesterday

- .github/workflows fix project name 2 weeks ago
- nuget fix: csproj path 2 weeks ago
- src Place the .IsNumer... yesterday
- .gitattributes add .gitattributes 2 years ago
- .gitignore update .gitignore 3 weeks ago
- LICENSE Initial commit 5 years ago
- README.md Support only .NET 8 2 weeks ago

README MIT license

### FastEnum

FastEnum is **extremely fast** enum utilities for C#/.NET. It's much faster than .NET. Provided methods are all achieved **zero allocation** and are designed easy to use like `System.Enum`. This library is quite useful to significantly improve your performance because enum is really popular feature.

Releases 19

v1.8.0 Latest on Jun 29, 2022

+ 18 releases

Packages

No packages published

<https://github.com/xin9le/FastEnum>

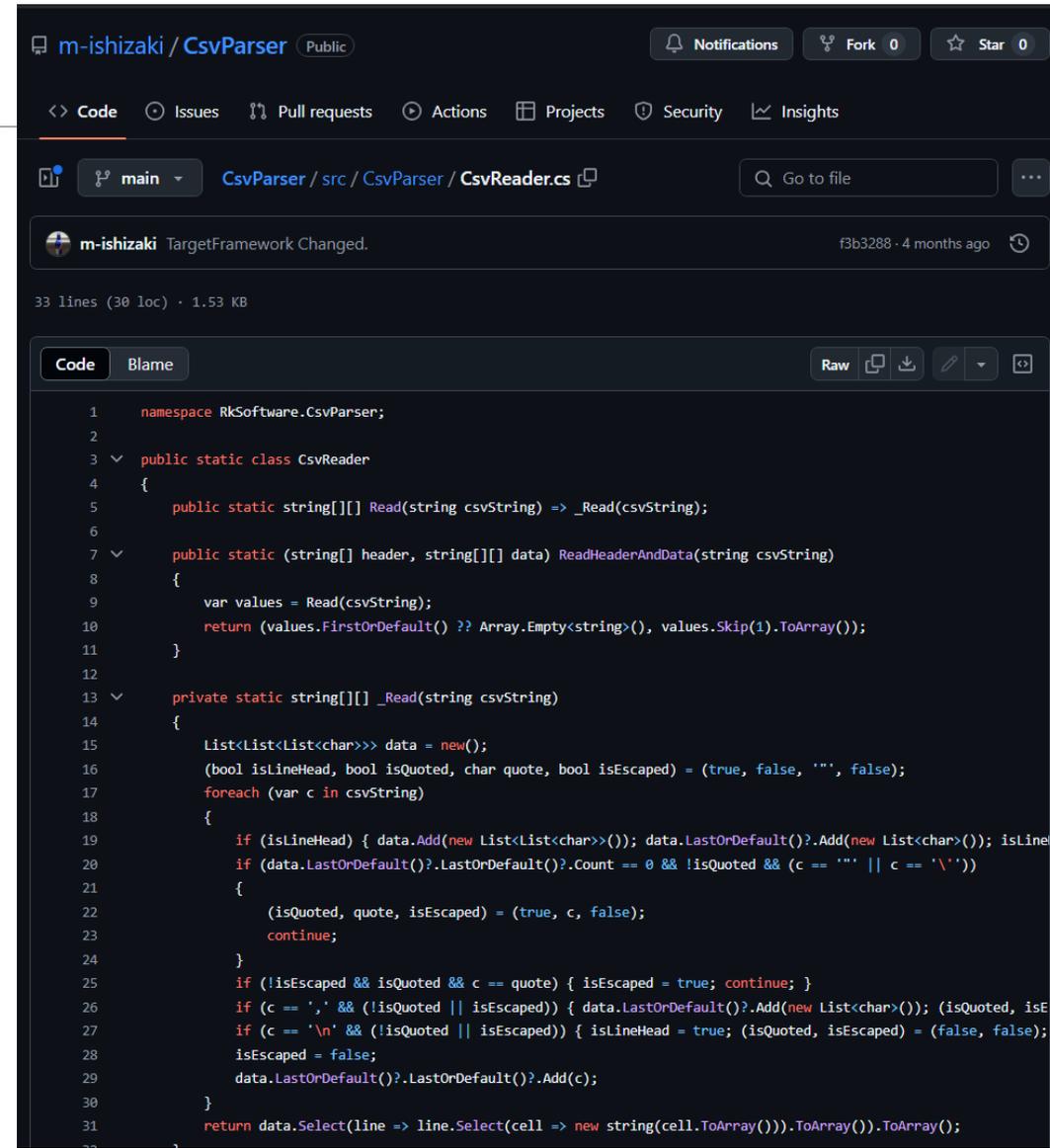
# さっきの CSV Parser

敷居を下げるのが割と得意です。

たったの **54 行** のソースコードでオープンしています。

ダウンロードして  
ソースコード部分  
(2,300 文字・54 行)

サイズ:	3.61 KB (3,702 バイト)
ディスク上のサイズ:	12.0 KB (12,288 バイト)
内容:	ファイル数: 3、フォルダー数: 0



```
1 namespace RkSoftware.CsvParser;
2
3 public static class CsvReader
4 {
5     public static string[][] Read(string csvString) => _Read(csvString);
6
7     public static (string[] header, string[][] data) ReadHeaderAndData(string csvString)
8     {
9         var values = Read(csvString);
10        return (values.FirstOrDefault() ?? Array.Empty<string>(), values.Skip(1).ToArray());
11    }
12
13    private static string[][] _Read(string csvString)
14    {
15        List<List<List<char>>> data = new();
16        (bool isLineHead, bool isQuoted, char quote, bool isEscaped) = (true, false, '', false);
17        foreach (var c in csvString)
18        {
19            if (isLineHead) { data.Add(new List<List<char>>()); data.LastOrDefault()?.Add(new List<char>()); isLine
20            if (data.LastOrDefault()?.LastOrDefault()?.Count == 0 && !isQuoted && (c == '' || c == '\\'))
21            {
22                (isQuoted, quote, isEscaped) = (true, c, false);
23                continue;
24            }
25            if (!isEscaped && isQuoted && c == quote) { isEscaped = true; continue; }
26            if (c == ',' && (!isQuoted || isEscaped)) { data.LastOrDefault()?.Add(new List<char>()); (isQuoted, isE
27            if (c == '\n' && (!isQuoted || isEscaped)) { isLineHead = true; (isQuoted, isEscaped) = (false, false);
28            isEscaped = false;
29            data.LastOrDefault()?.LastOrDefault()?.Add(c);
30        }
31        return data.Select(line => line.Select(cell => new string(cell.ToArray())).ToArray()).ToArray();
32    }
33 }
```

1. (分量：03 ページ) ソースオープンとは
2. (分量：05 ページ) 人はなぜブログを書くのか
3. (分量：02 ページ) とはいえそんな分量は書けない
4. (分量：04 ページ) **今日のために Pleasanter に AI を組み込んでみた**
5. (分量：01 ページ) まとめ

---

## 目次

# AI を使うコードを GitHub で公開

つまり

皆さん Pleasanter に AI を組み込めるといことです

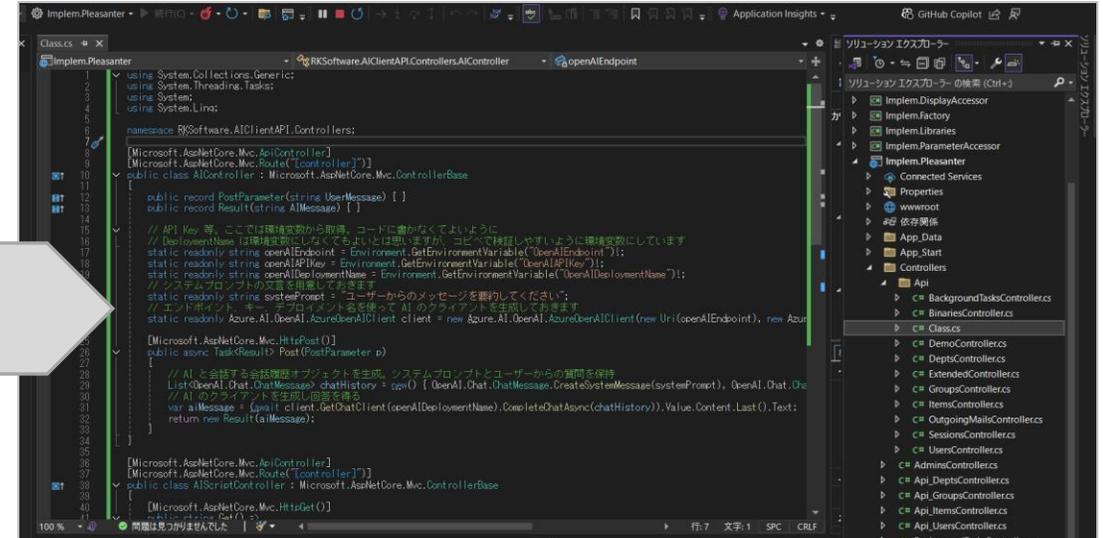
本当は皆さんがソースコードを見なくても使えるようにしたかったのですが、時間切れ……

```
1 namespace RKSoftware.AIClientAPI.Controllers;
2
3 [Microsoft.AspNetCore.Mvc.ApiController]
4 [Microsoft.AspNetCore.Mvc.Route("[controller]")]
5 public class AIController : Microsoft.AspNetCore.Mvc.ControllerBase
6 {
7     public record PostParameter(string UserMessage) { }
8     public record Result(string AIMessage) { }
9
10    // API Key 等。ここでは環境変数から取得。コードに書かなくてよいように
11    // DeploymentName は環境変数にしないともよいとは思いますが、コピペで検証しやすいように環境変数にしています
12    static readonly string openAIEndpoint = Environment.GetEnvironmentVariable("OpenAIEndpoint");
13    static readonly string openAIAPIKey = Environment.GetEnvironmentVariable("OpenAIAPIKey");
14    static readonly string openAIDeploymentName = Environment.GetEnvironmentVariable("OpenAIDeploymentName");
15    // システムプロンプトの文言を用意しておきます
16    static readonly string systemPrompt = "ユーザーからのメッセージを要約してください";
17    // エンドポイント、キー、デプロイメント名を使って AI のクライアントを生成しておきます
18    static readonly Azure.AI.OpenAI.AzureOpenAIClient client = new Azure.AI.OpenAI.AzureOpenAIClient(new Uri(openAIEndpoint));
19
20    [Microsoft.AspNetCore.Mvc.HttpPost()]
21    public async Task<Result> Post(PostParameter p)
22    {
23        // AI と会話する会話履歴オブジェクトを生成。システムプロンプトとユーザーからの質問を保持
24        List<OpenAI.Chat.ChatMessage> chatHistory = new() { OpenAI.Chat.ChatMessage.CreateSystemMessage(systemPrompt), OpenAI.Chat.ChatMessage.CreateUserMessage(p.UserMessage) };
25        // AI のクライアントを生成し回答を得る
26        var aiMessage = (await client.GetChatClient(openAIDeploymentName).CompleteChatAsync(chatHistory)).Value.Content.Last();
27        return new Result(aiMessage);
28    }
29 }
30
31 [Microsoft.AspNetCore.Mvc.ApiController]
32 [Microsoft.AspNetCore.Mvc.Route("[controller]")]
33 public class AIScriptController : Microsoft.AspNetCore.Mvc.ControllerBase
34 {
35     [Microsoft.AspNetCore.Mvc.HttpGet()]
36     public string Get() => ""
37     {
38         function addAIClientAPI(idButton, idTextarea, site) {
39             document.getElementById(idButton).setAttribute('onclick', `aiClientAPI('${idTextarea}', site);return false;`);
40         }
41     }
42 }
```

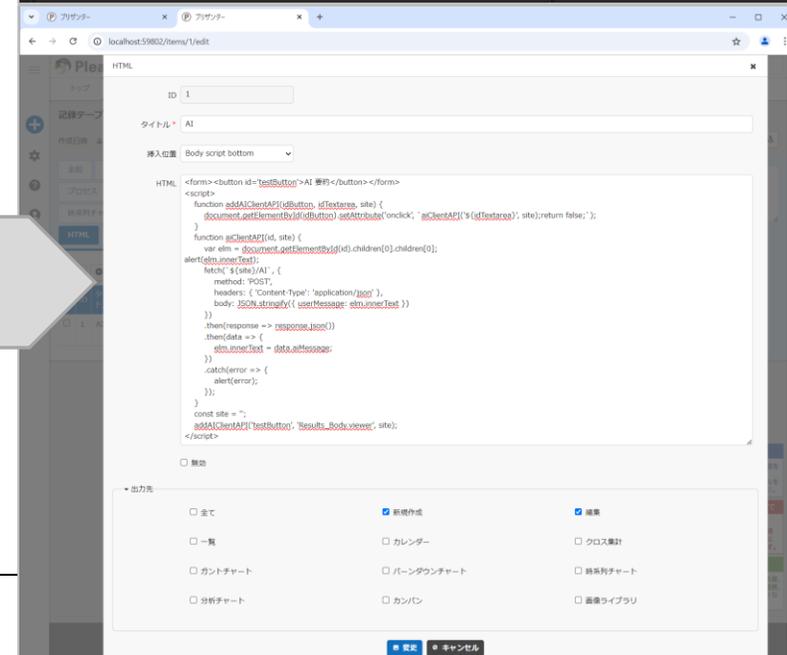
# ソースコード テーブルの管理

## Pleasanter のソースにファイルを追加

C# コードのソースコードファイルとして読まれるところならどこにでも



テーブルの管理でボタンと↑で追加した機能呼び出すスクリプトを追加



# ソースコード貼り付けページ

今は読まなくて大丈夫です。後でじっくり読んでください

```
using System.Collections.Generic;
using System.Threading.Tasks;
using System;
using System.Linq;

namespace RKSoftware.AIClientAPI.Controllers;

[Microsoft.AspNetCore.Mvc.ApiController]
[Microsoft.AspNetCore.Mvc.Route("[controller]")]
public class AIController : Microsoft.AspNetCore.Mvc.ControllerBase
{
    public record PostParameter(string UserMessage) {}
    public record Result(string AIMessage) {}

    // API Key 等。ここでは環境変数から取得。コードに書かなくてよいように
    // DeploymentName は環境変数にしなくてもよいとは思いますが、コピペで検証しやすいように環境変数にしています
    static readonly string openAIEndpoint = Environment.GetEnvironmentVariable("OpenAIEndpoint");
    static readonly string openAIAPIKey = Environment.GetEnvironmentVariable("OpenAIAPIKey");
    static readonly string openAIDeploymentName = Environment.GetEnvironmentVariable("OpenAIDeploymentName");
    // システムプロンプトの文言を用意しておきます
    static readonly string systemPrompt = "ユーザーからのメッセージを要約してください";
    // エンドポイント、キー、デプロイメント名を使って AI のクライアントを生成しておきます
    static readonly Azure.AI.OpenAI.AzureOpenAIClient client = new Azure.AI.OpenAI.AzureOpenAIClient(new Uri(openAIEndpoint),
    new Azure.AzureKeyCredential(openAIAPIKey));

    [Microsoft.AspNetCore.Mvc.HttpPost()]
    public async Task<Result> Post(PostParameter p)
    {
        // AI と会話する会話履歴オブジェクトを生成。システムプロンプトとユーザーからの質問を保持
        List<OpenAI.Chat.ChatMessage> chatHistory = new() { OpenAI.Chat.ChatMessage.CreateSystemMessage(systemPrompt),
        OpenAI.Chat.ChatMessage.CreateUserMessage(p.UserMessage) };
        // AI のクライアントを生成し回答を得る
        var aiMessage = (await
        client.GetChatClient(openAIDeploymentName).CompleteChatAsync(chatHistory)).Value.Content.Last().Text;
        return new Result(aiMessage);
    }
}
```

```
<form><button id='testButton'>AI 要約</button></form>
<script>
    function addAIClientAPI(idButton, idTextarea, site) {
        document.getElementById(idButton).setAttribute('onclick', `aiClientAPI('${idTextarea}', site);return false;`);
    }
    function aiClientAPI(id, site) {
        var elm = document.getElementById(id).children[0].children[0];
        fetch(`${site}/AI`, {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ userMessage: elm.innerText })
        })
        .then(response => response.json())
        .then(data => {
            elm.innerText = data.aiMessage;
        })
        .catch(error => {
            alert(error);
        });
    }
    const site = "";
    addAIClientAPI('testButton', 'Results_Body.viewer', site);
</script>
```

# 実行結果

☰ Pleasanter

🔗 トップ 記録テーブル

+

📄

⚙️

?

👤

テスト

作成日時 Administrator 2024/09/16 月 13:11:33 10 時間前 更新日時 Administrator 2024/09/16 月 23:00:12 7 分前

全般 変更履歴の一覧 レコードのアクセス制御

ID 2 バージョン 1

タイトル\* テスト

内容  
コミット履歴が恥ずかしい。でも GitHub で PR 出したい！メモ。技術 Visual Studio。自分のコミット履歴が恥ずかしい、でも C# Tokyo のお題にチャレンジして PR したい！そう考えていますね。大丈夫です、コミット履歴などきれいにする必要はありません。何なら既存を破壊するような PR を出しても大丈夫です！最後に整っていればそれでよいのですし、マージ後に最後に整えますので。それでもやっぱり恥ずかしい。わかります。私も恥ずかしいです。なので簡単にコミットをきれいにしつつ PR する手順に必要なコマンド等をメモしておきます。私自身が覚えられないので、スニペットとしておいておきたいので。

📎 📎

状況 未着手 管理者 Administrator 担当者 Administrator

新バージョンとして保存

AI 要約

1. (分量：03 ページ) ソースオープンとは
2. (分量：05 ページ) 人はなぜブログを書くのか
3. (分量：02 ページ) とはいえそんな分量は書けない
4. (分量：04 ページ) 今日のために Pleasanter に AI を組み込んでみた
5. **(分量：01 ページ) まとめ**

---

## 目次

# まとめ

---

- 学んだことを GitHub に置いていけば
- **Web 検索で**いつでもどこでも参照可能
- 自分のために自分の知識を OSS に
- どこに居ても、**どこに行っても Pleasanter** に自分機能を追加

ありがとうございました。

---

石崎 充良