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/

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？

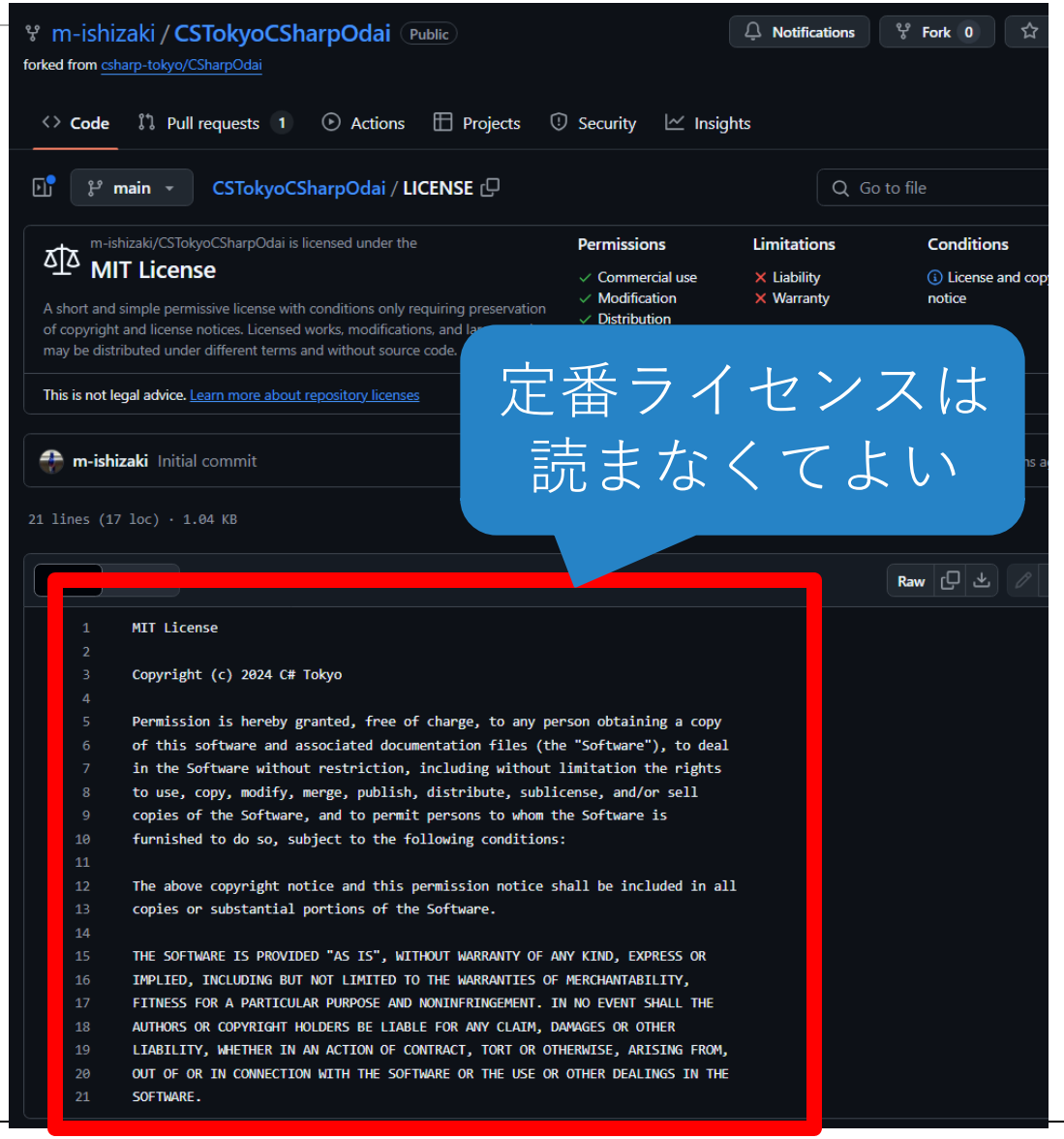
OSS にしてください

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

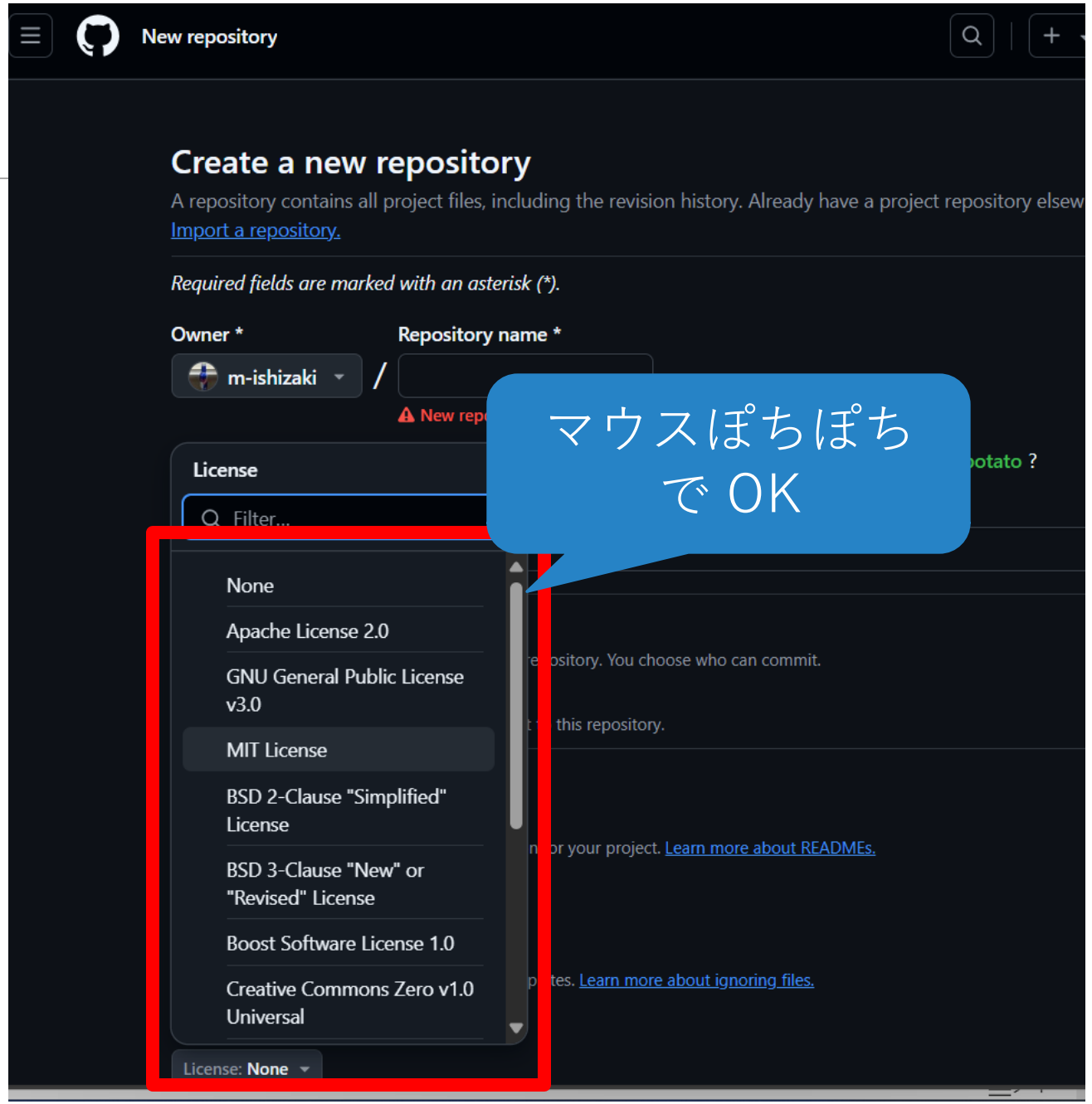
軍事利用が可能でないならば OSS ではありません



https://github.com/m-ishizaki/CSTokyoCSharpOdai/blob/main/LICENSE

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

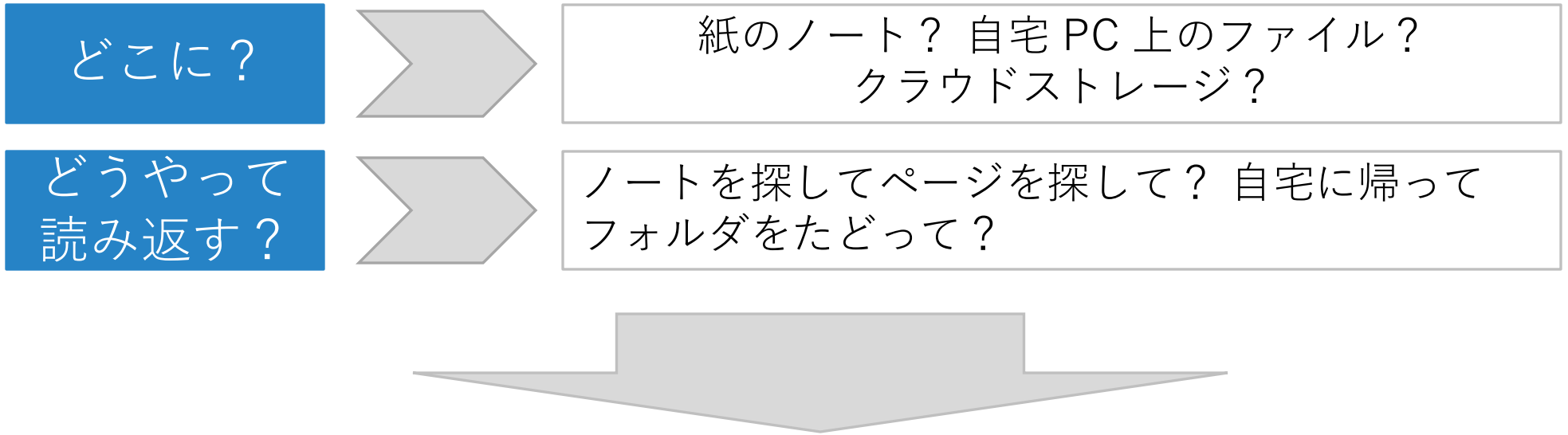


https://github.com/new

目次

人は忘れる生き物だから

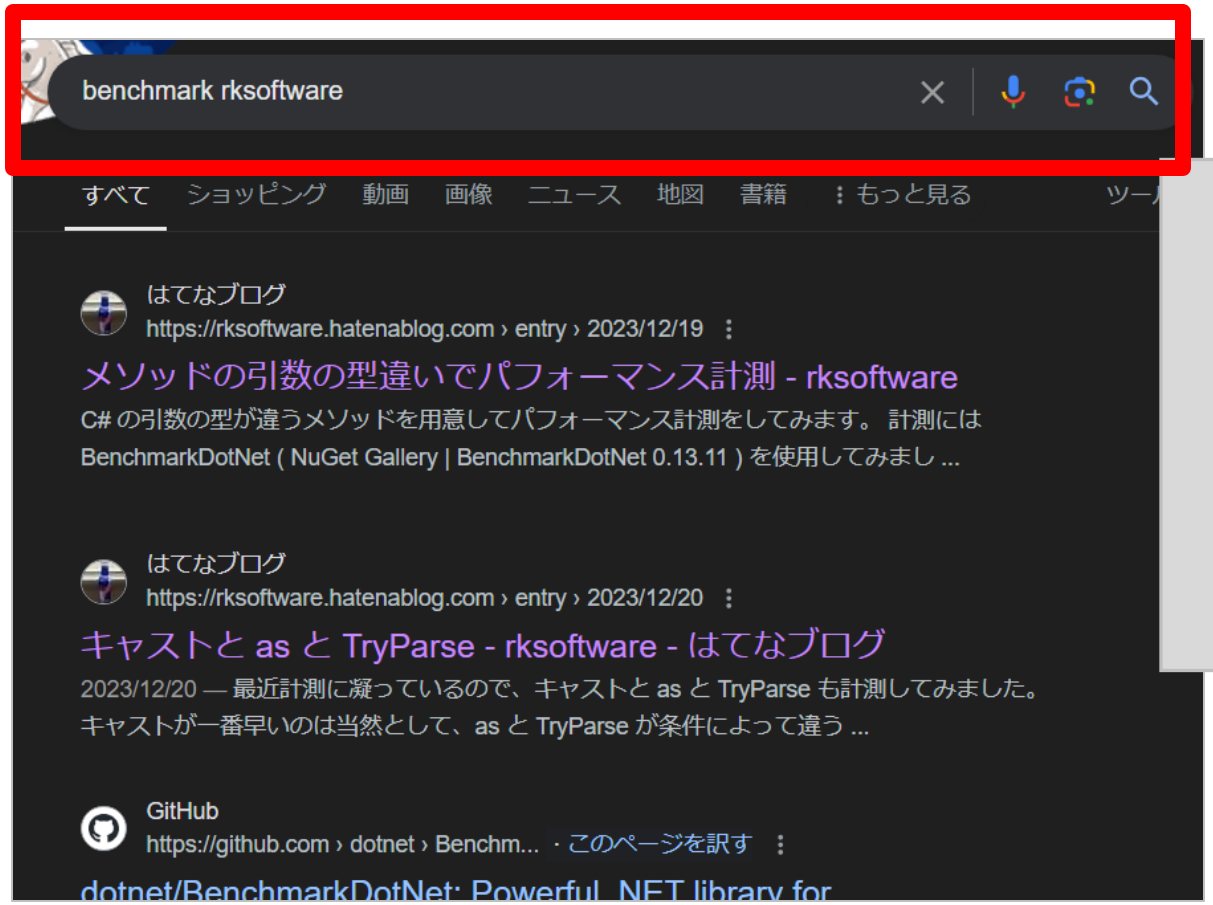
学んだことは書き留めておかねばなりません



いつでも、どこでも Web 検索で読み返せたら最強なのでは！

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 型のものと、int 型のものと、string 型のものです。

```
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();
```

https://www.google.com/search
https://rksoftware.hatenablog.com/entry/2023/12/19/010000

不十分・わかりにくい

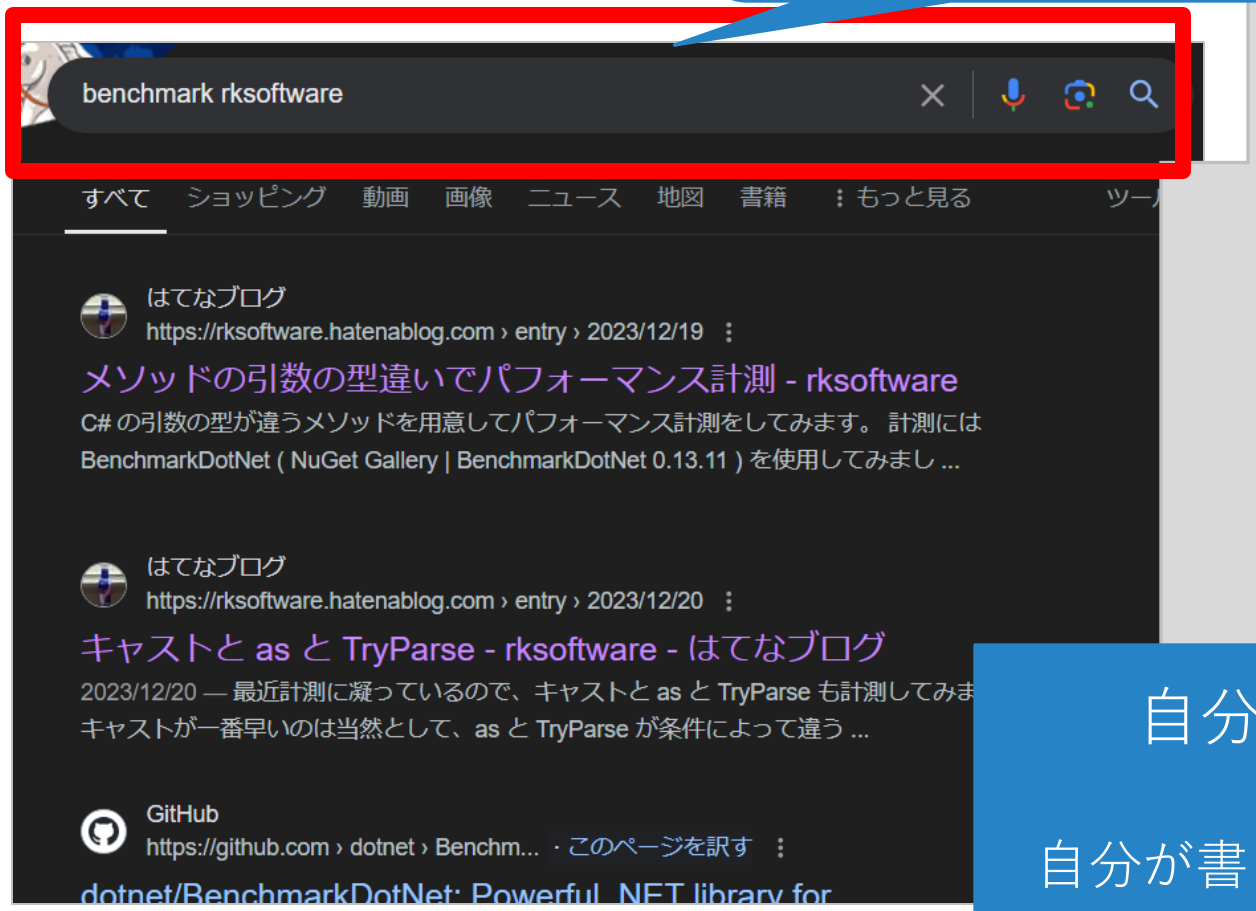
調べものをしていて、良さそうな記事を見つけました。喜んで読んでみると

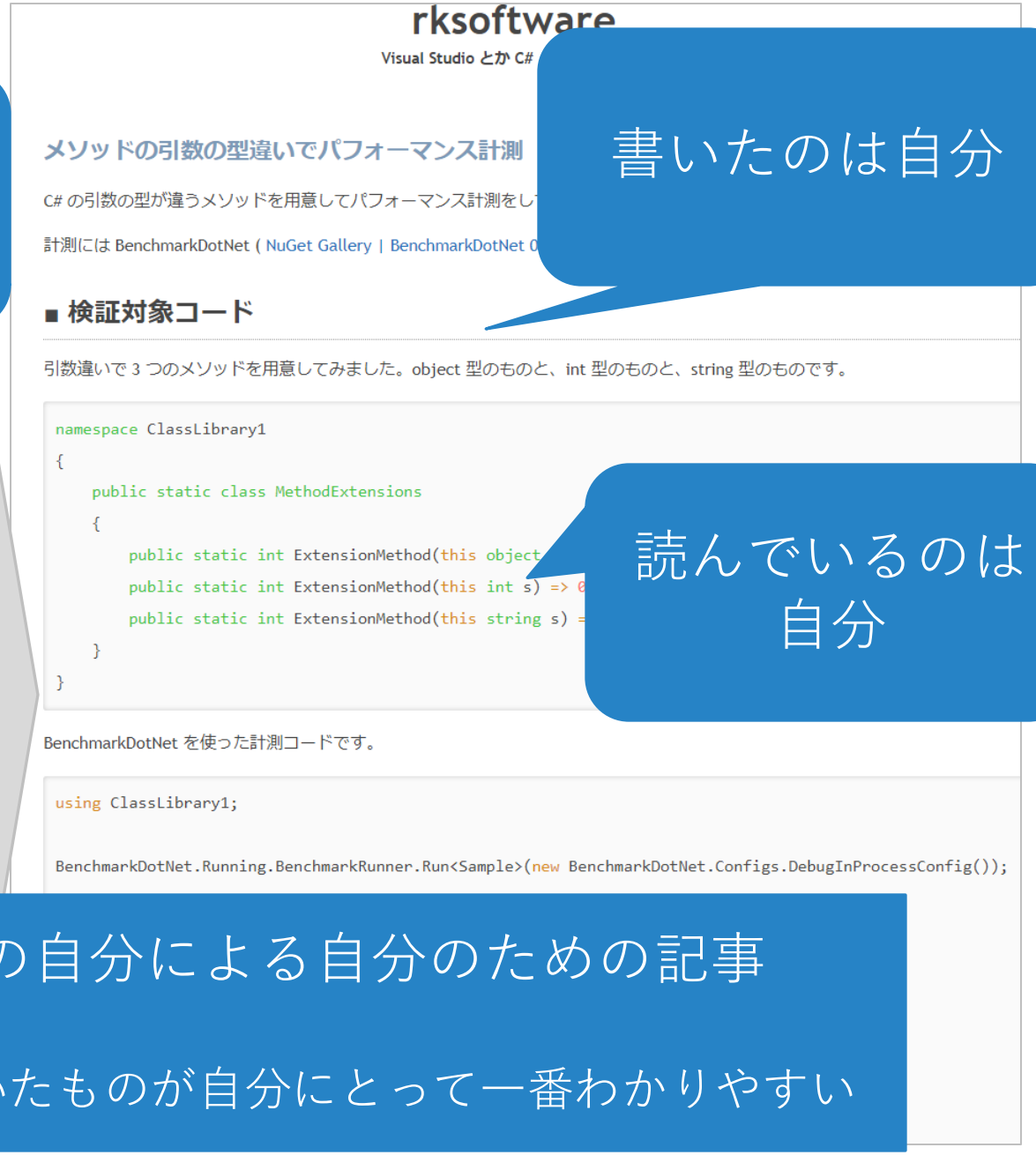
記事を書いた **本人にとって** 十分でわかりやすい

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

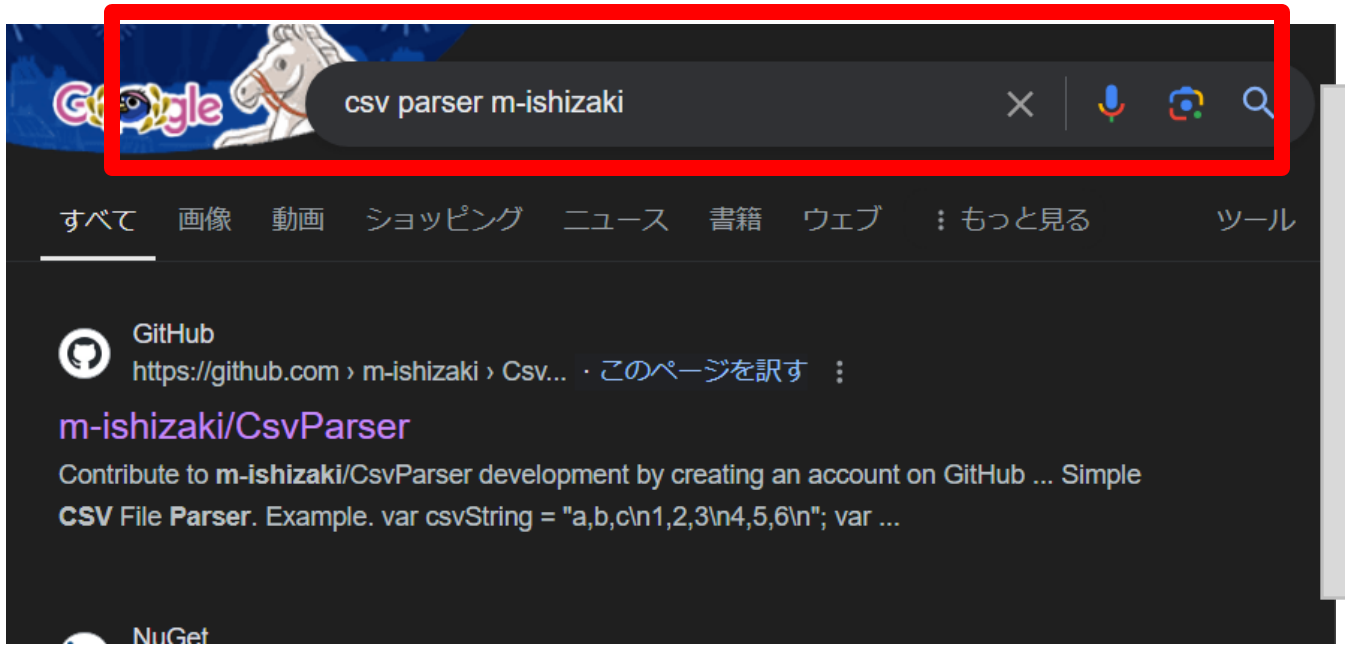
自分の自分による自分のための記事

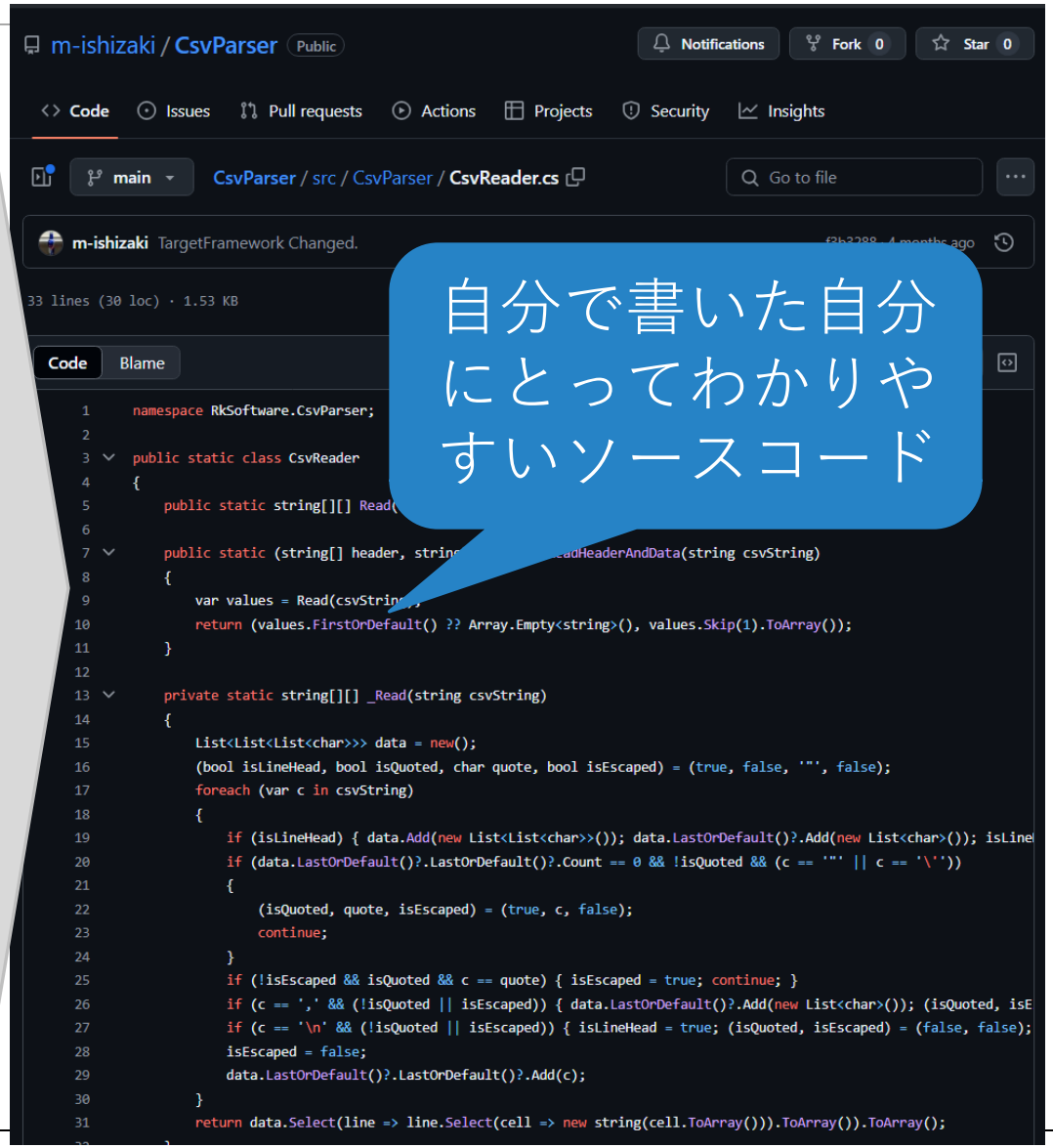
自分が書いたものが自分にとって一番わかりやすい

https://www.google.com/search
https://rksoftware.hatenablog.com/entry/2023/12/19/010000

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？





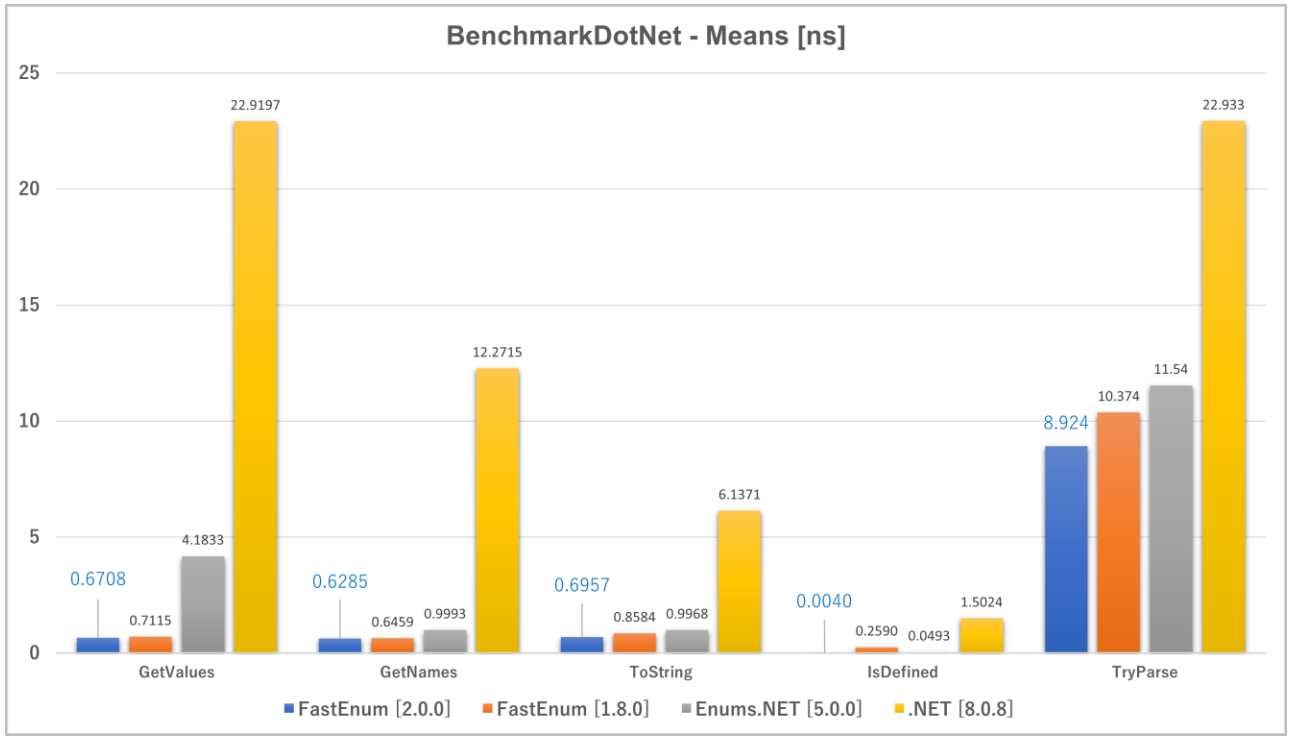
https://www.google.com/search
https://github.com/m-ishizaki/CsvParser/blob/main/src/CsvParser/CsvReader.cs

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

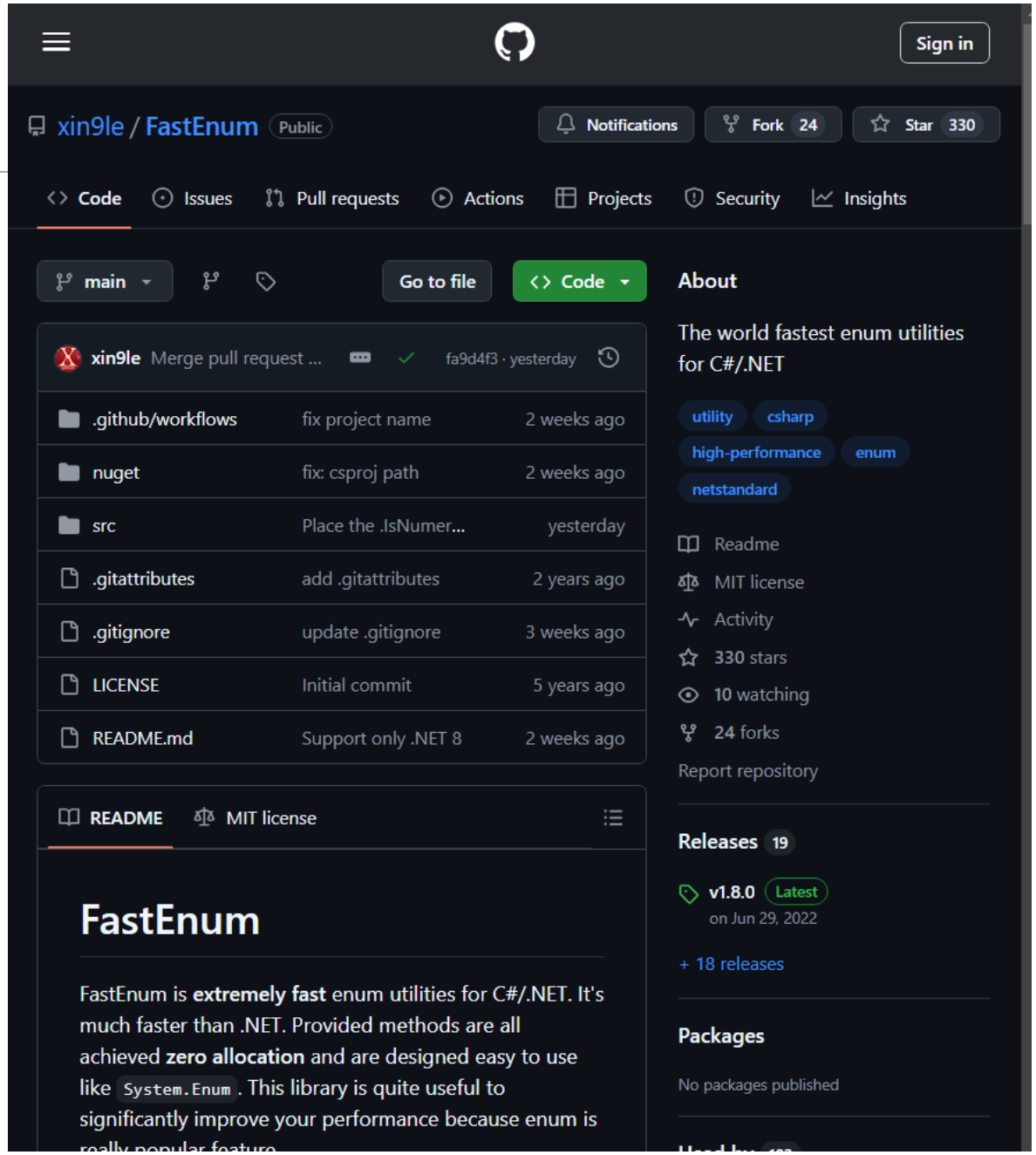
大丈夫です

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、フォルダー数: 1



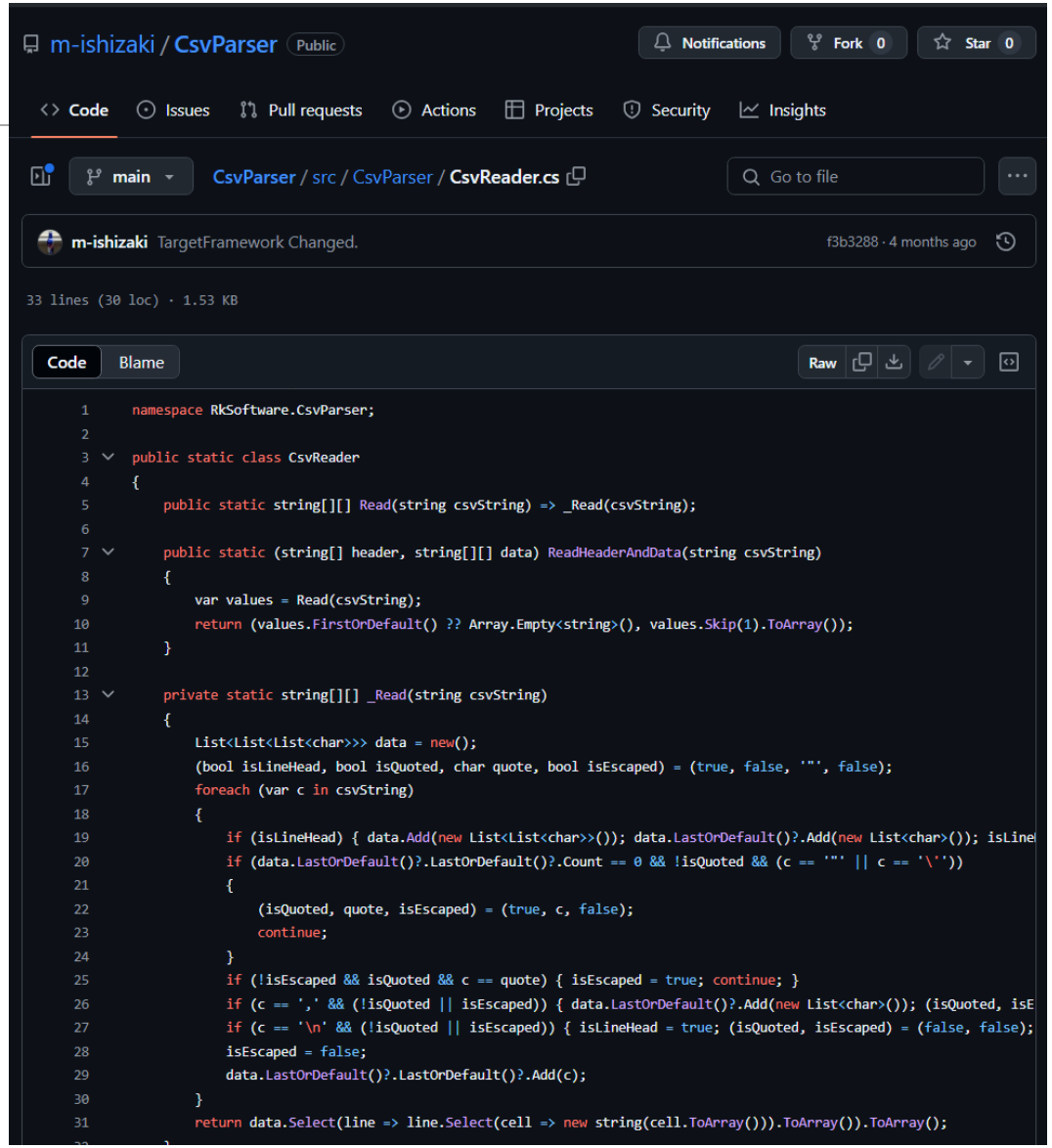
※ Pleasanter には 1 ファイル 5 百万文字 5MB あり

https://github.com/xin9le/FastEnum

さっきの CSV Parser

敷居を下げることが割と得意です。

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

https://github.com/m-ishizaki/CsvParser/blob/main/src/CsvParser/CsvReader.cs

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

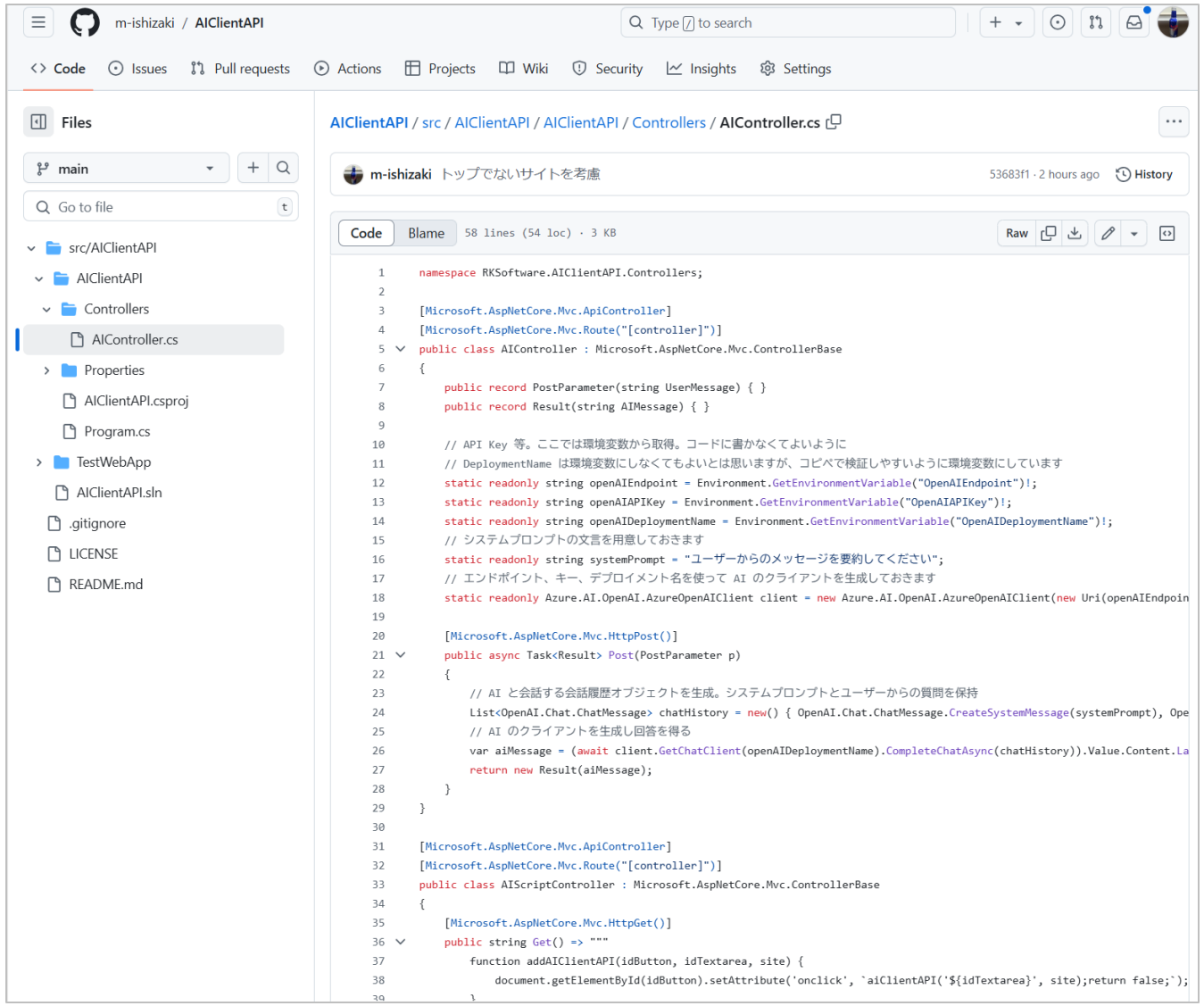
目次

AI を使うコードを GitHub で公開

つまり

皆さん Pleasanter に AI を組み込めるということです

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……

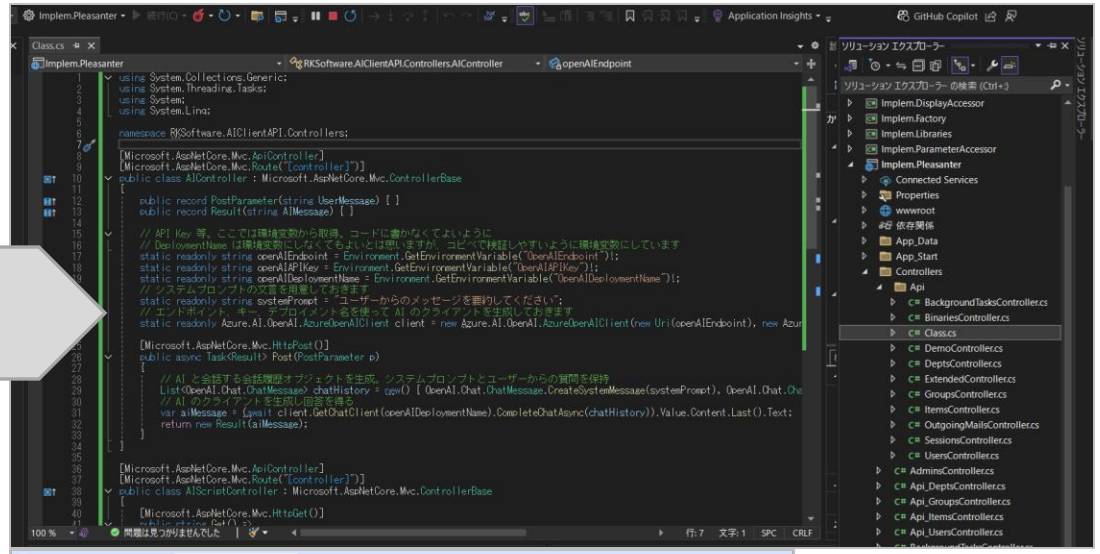


https://github.com/m-ishizaki/AIClientAPI
https://github.com/m-ishizaki/OpenAIChatSample/

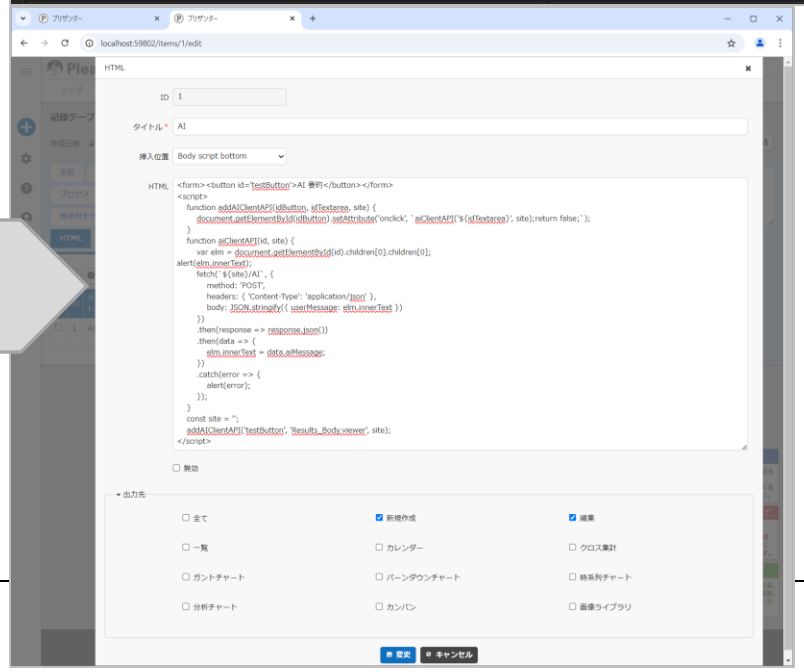
ソースコード テーブルの管理

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↑で追加した機能を呼び出すスクリプトを追加



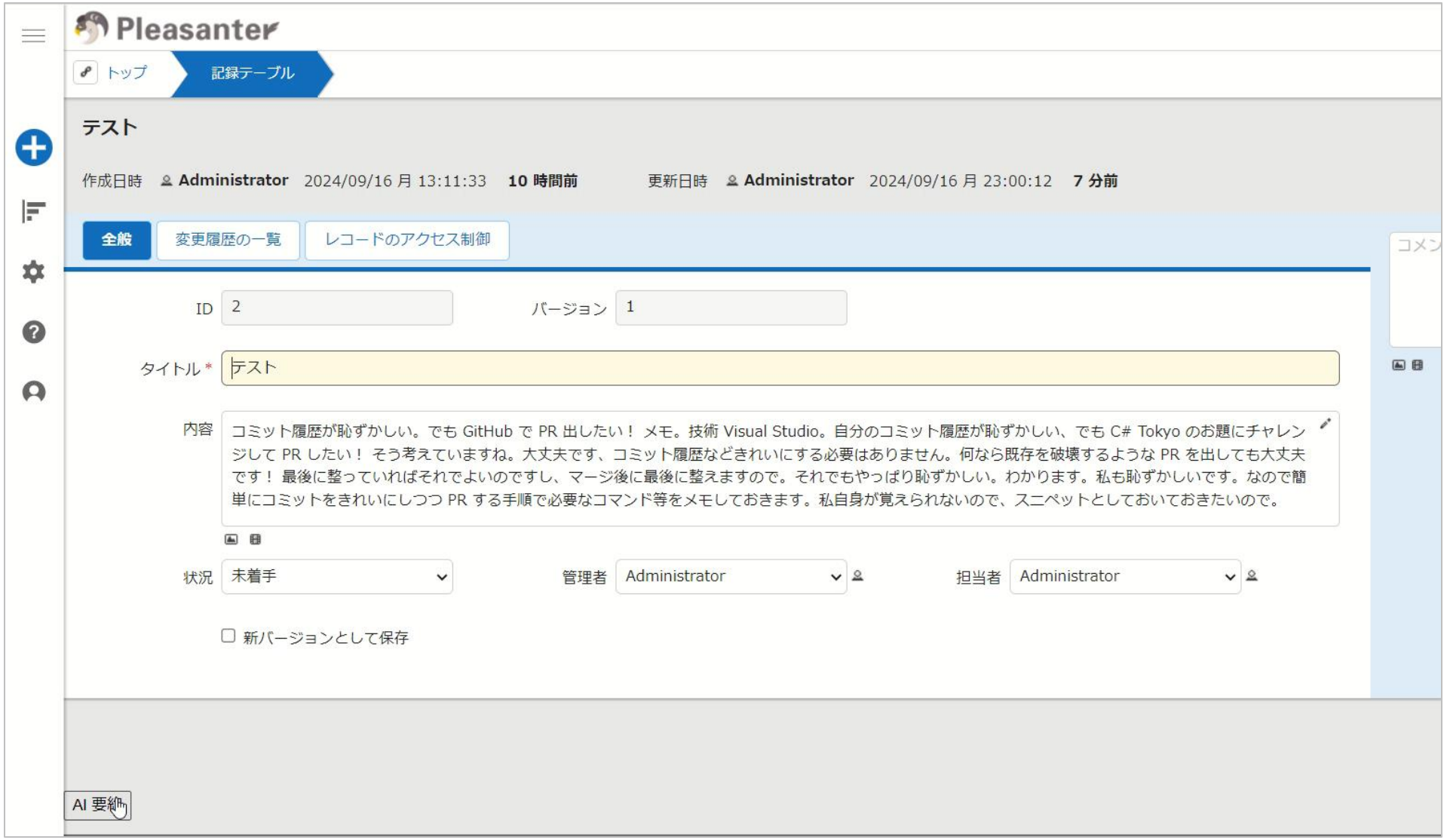
ソースコード貼り付けページ

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 する手順で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

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