# 情報発信をしよう

# &

# アプリに何もしないでアプリに AI を組み込む

---

【登壇者募集中】.NET Conf 2024 後！ C# Tokyo カンファレンス

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/

# 情報発信をしよう

# 【登壇者募集中】









https://csharp-tokyo.connpass.com/event/334045/
https://www.youtube.com/@CTokyo-wj8fv

目次

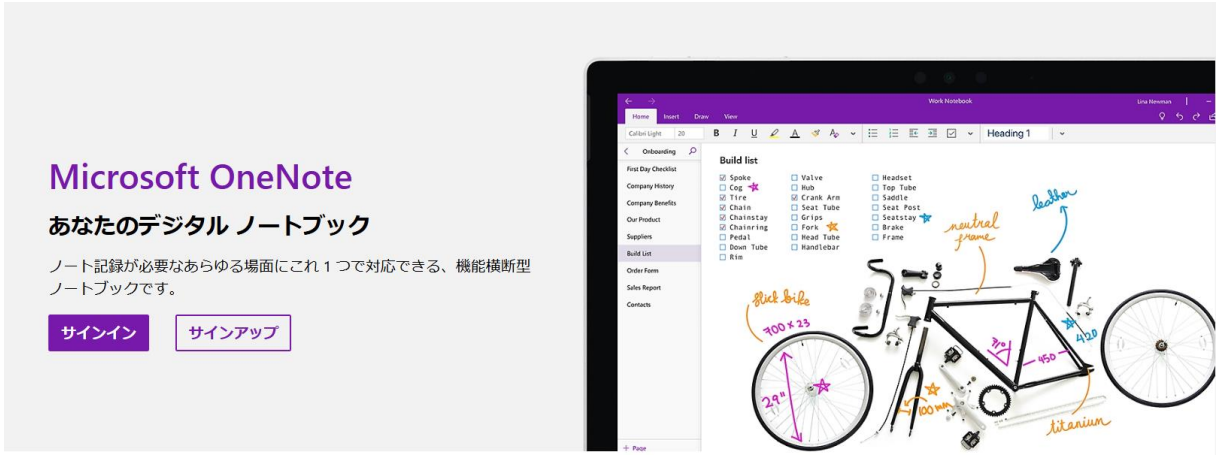
# OneNote は素晴らしい製品です

人は学ぶ生き物 → しかし → ~~忘れてしまう生き物~~

例えば

**OneNote**

https://www.microsoft.com/ja-jp/microsoft-365/onenote/digital-note-taking-app




しかし大きな課題も

# おうち vs 職場

学ぶのは自分の環境

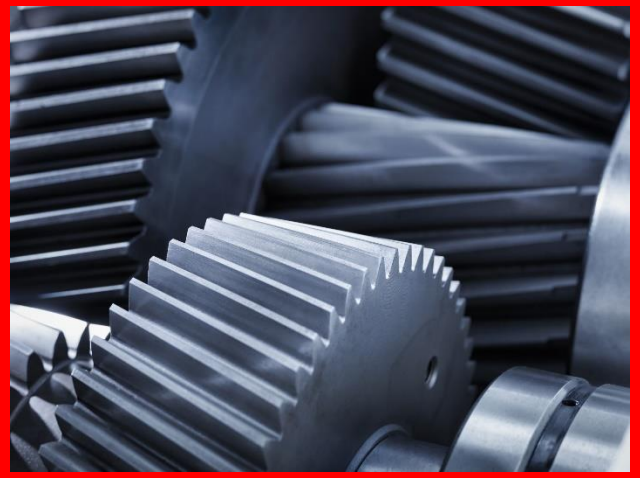
活用は職場

個人のアカウント（環境）

OneNote

参照（コピペ）
できない

職場の環境



仕事 PC

# いつでも、どこからでも

学ぶのは自分の環境

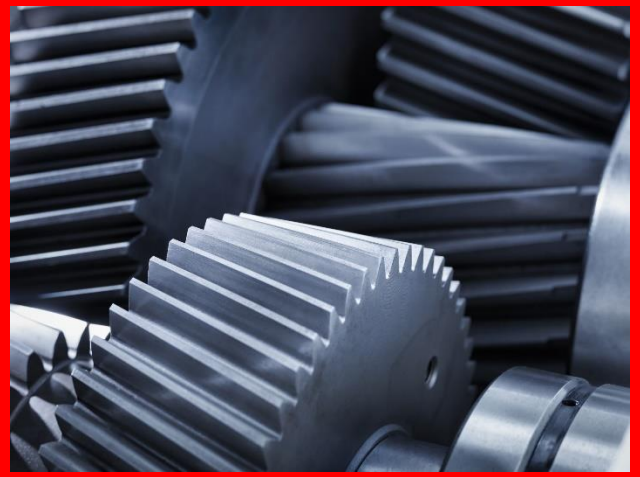
活用は職場

インターネットに公開

ブログ

参照（コピペ）できる

職場の環境



仕事 PC

# 職場は変わる恐れがある

自分の環境

インターネットに公開

ブログ

ずっと
メンテ可能

職場

職場の環境

ブログ

メンテ不可

転

職場の環境

ブログ

---

目次


職場ブログを
書いてほしい時は

# 自分で学ぶ隙を与えない

学ぶのは自分の環境

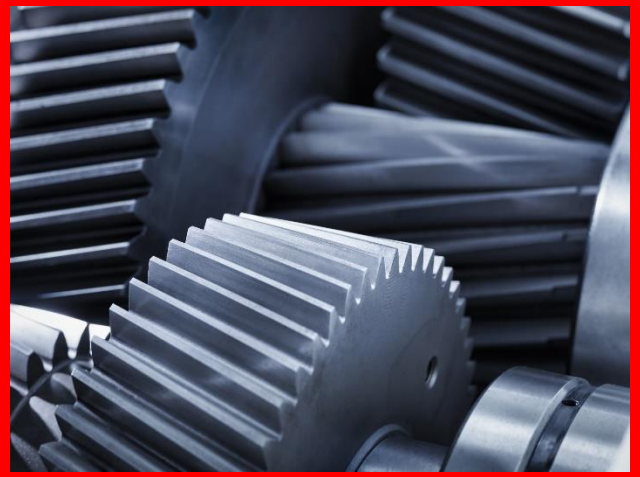
インターネットに公開

ブログ

職場での学びがファストだと自分ブログに書かない

学ぶのは職場

職場の環境



仕事 PC

# 時系列

新しいものが登場

調査

ブログ

調査

ブログ

調査

ブログ

# 時系列

新しいものが登場

日本時間深夜に公開

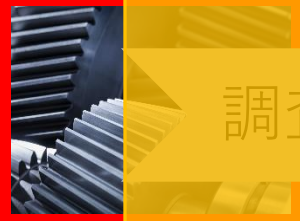
翌日の業後

調査

ブログ

調査

ブログ



調査

ブログ

---

目次


自分が良ければよい

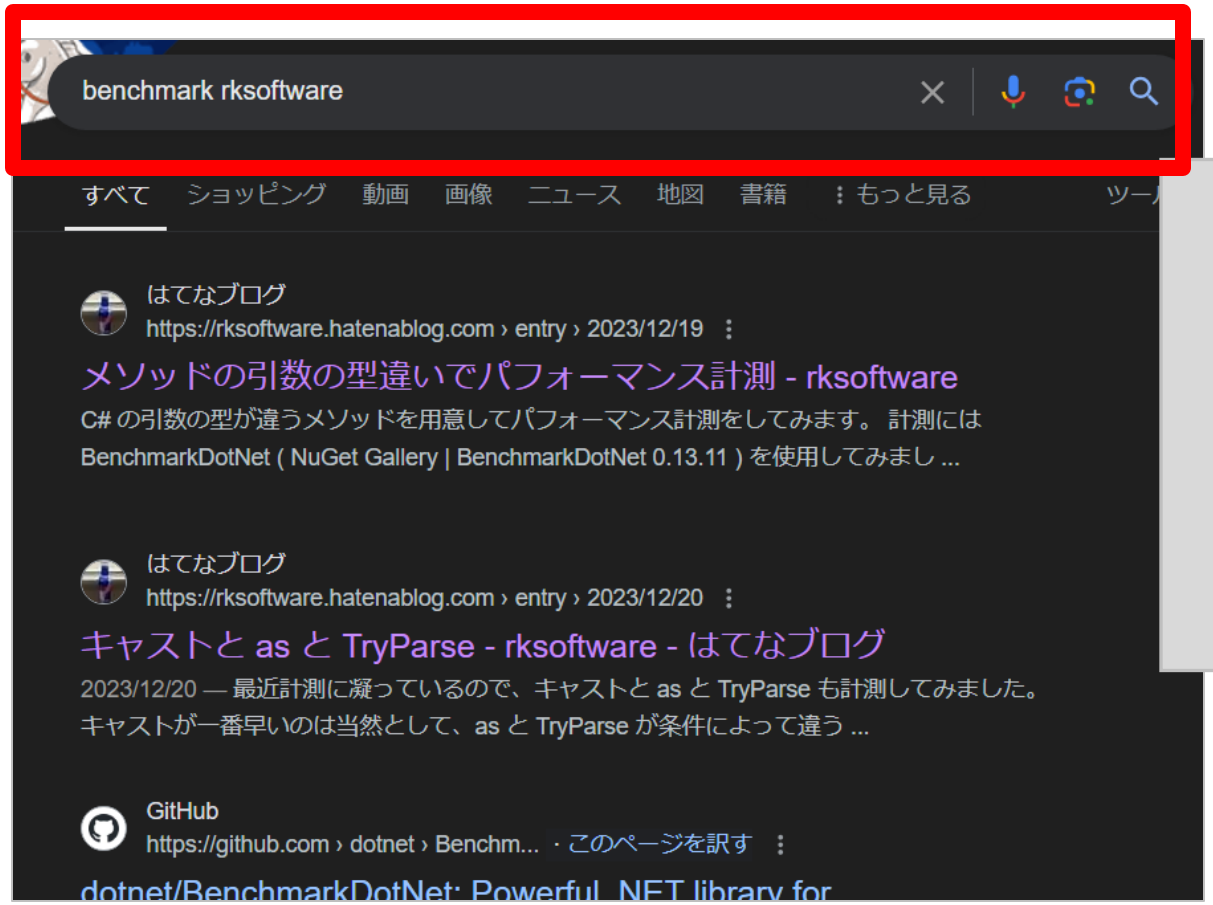
# 不十分・わかりにくい

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。

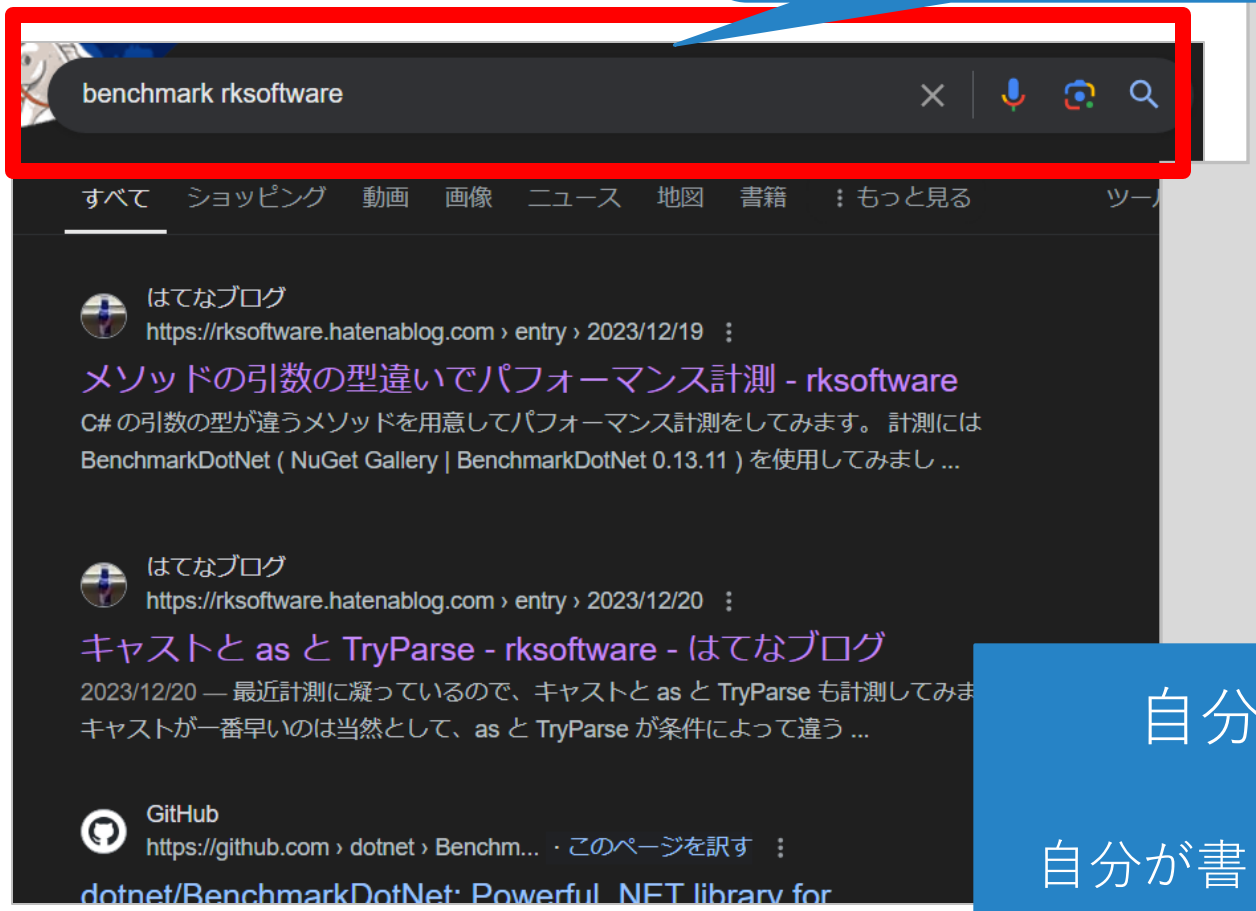
```
using ClassLibrary1;

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();
```

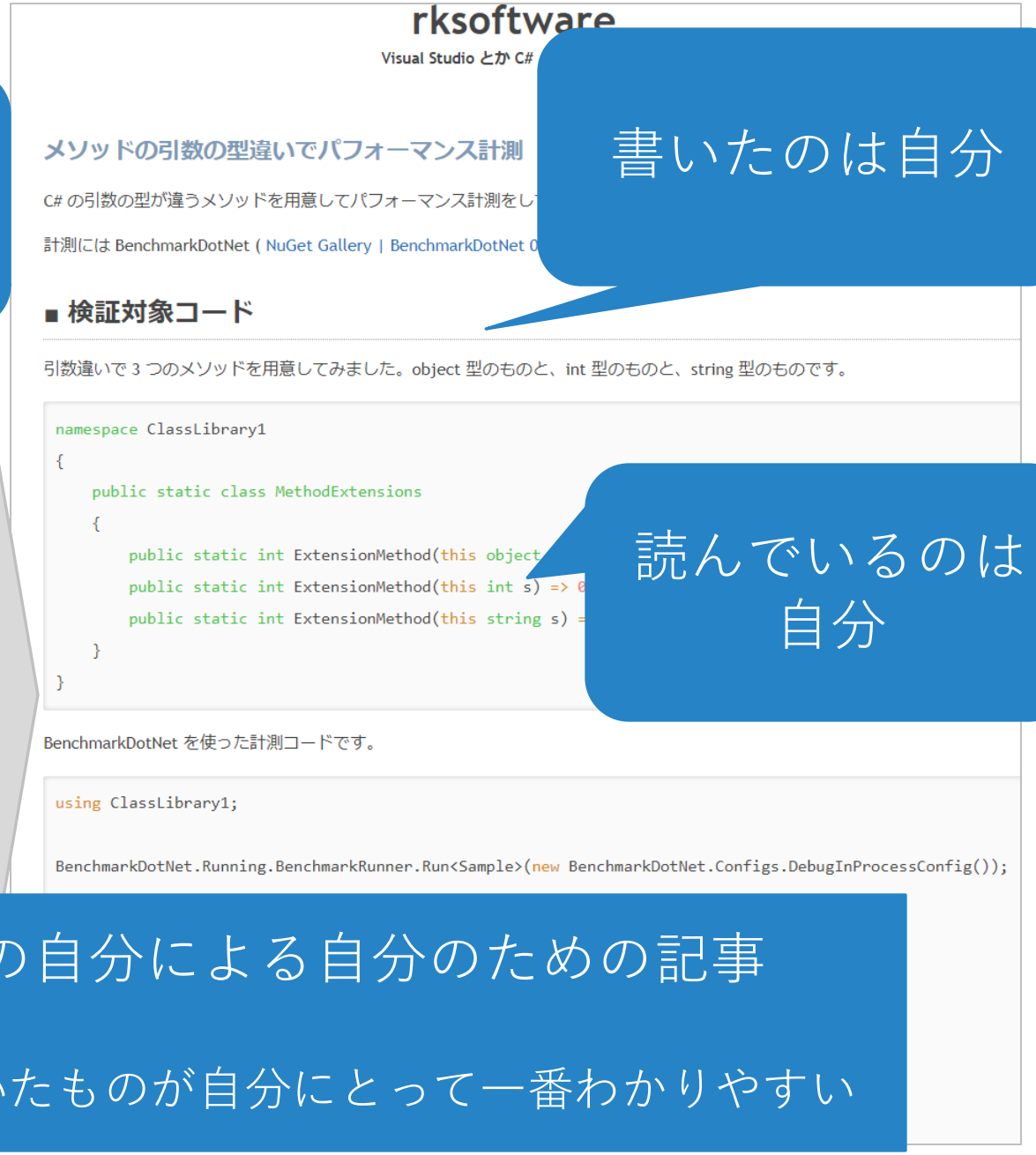
https://www.google.com/search
https://rksoftware.hatenablog.com/entry/2023/12/19/010000

# 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

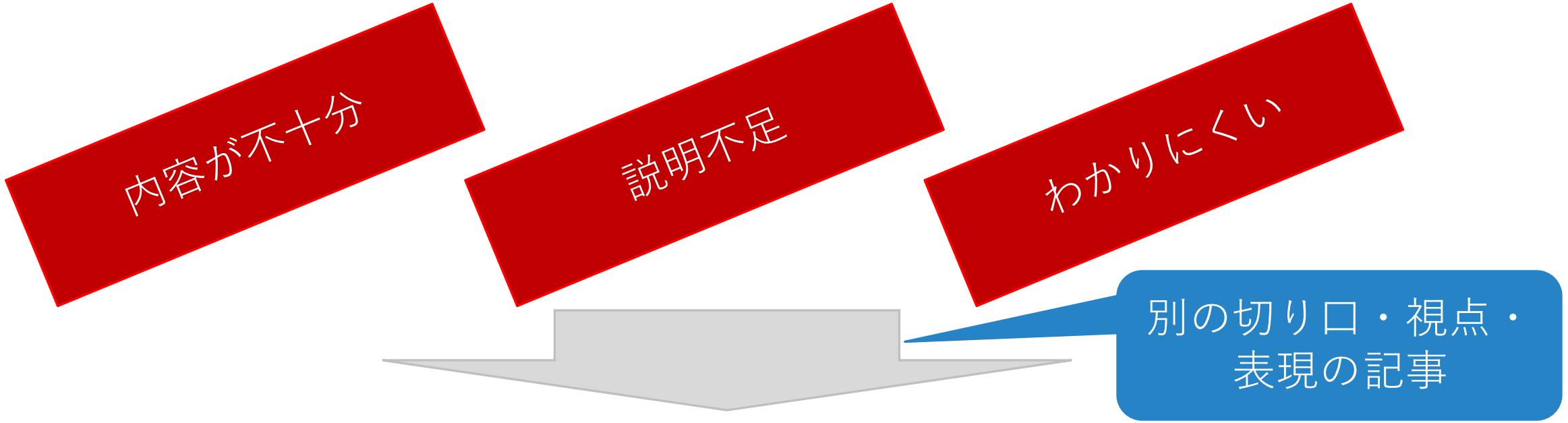
自分の自分による自分のための記事

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

# 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

自分に近い状況の人にとっても
わかりやすいかもしれない

目次


締め切り駆動
情報発信

# 締め切り駆動　情報発信


動画公開


スライド
公開



閲覧可能

https://csharp-tokyo.connpass.com/event/334045/
https://www.youtube.com/@CTokyo-wj8fv

# 【登壇者募集中】


https://csharp-tokyo.connpass.com/event/334045/
https://www.youtube.com/@CTokyo-wj8fv


アプリに何もしないで
アプリに AI を組み込む

目次

# 結論

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
var mvcBuilder = builder.Services.AddControllersWithViews();

// dll 読み込み
{
    var basePath = Path.Combine(Directory.GetParent(Assembly.GetEntryAssembly()!.Location!)!.Parent!.Parent!.Parent!.Parent!.FullName, "AIClientAPI\\bin");
    var files = Directory.GetFiles(basePath, "*.dll", SearchOption.AllDirectories).Select(dll => Assembly.LoadFrom(dll)).ToArray();
    foreach (var assembly in files)
    {
        mvcBuilder.AddApplicationPart(assembly);
    }
}
```

DLL 読み込み

コントローラーを追加
(コントローラーを探す場所に DLL を追加)

MvcCoreMvcBuilderExtensions.AddApplicationPart(IMvcBuilder, Assembly) メソッド

## 定義

名前空間: Microsoft.Extensions.DependencyInjection

アセンブリ: Microsoft.AspNetCore.Mvc.Core.dll

パッケージ: Microsoft.AspNetCore.App.Ref v9.0.0

の ApplicationPart 一覧 ApplicationParts に を追加します PartManager。

https://learn.microsoft.com/ja-jp/dotnet/api/microsoft.extensions.dependencyinjection.mvccoremvcbuilderextensions.addapplicationpart

# ASP.NET MVC の頃は？

## DLL を置くだけでコントローラーを追加できた

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
);
```

プロジェクトテンプレートで
生まれるコード

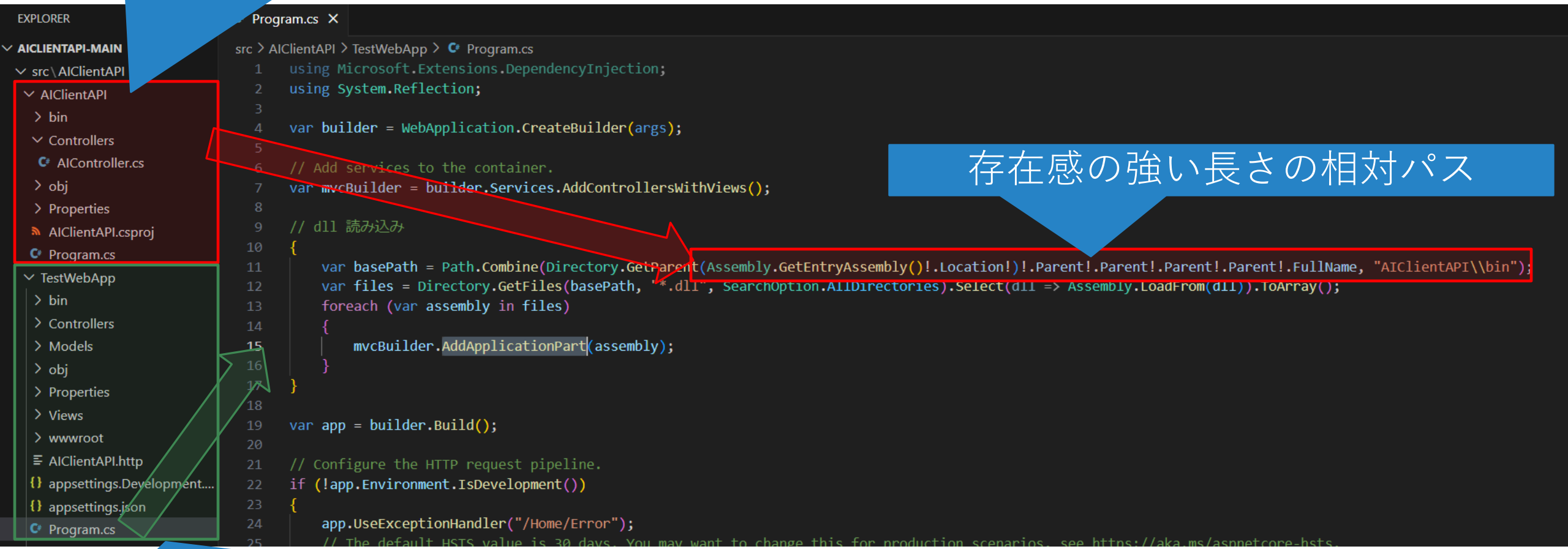
## 置いておく DLL の中身

```
public class LoadController : Controller
{
    public ActionResult Index()
    {
        return new ContentResult() { Content = "LoadController" + WebApplication1.Models.Dummy.Text };
    }
}
```

こういったクラスを含む
DLL を置いておくだけ

# デバッグを意識したプロジェクト構成

追加で置く DLL のプロジェクト




存在感の強い長さの相対パス

ASP.NET Core MVC プロジェクト

目次

# AI 機能を実装したコントローラー

```
namespace RKSoftware.AIClientAPI.Controllers;

[Microsoft.AspNetCore.Mvc.ApiController]
[Microsoft.AspNetCore.Mvc.Route("[controller]")]
0 references
public class AIController : Microsoft.AspNetCore.Mvc.ControllerBase
{
    1 reference
    public record PostParameter(string UserMessage) { }
    2 references
    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
    1 reference
    static readonly string openAIEndpoint = Environment.GetEnvironmentVariable("OpenAIEndpoint")!;
    1 reference
    static readonly string openAIAPIKey = Environment.GetEnvironmentVariable("OpenAIAPIKey")!;
    1 reference
    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
    1 reference
    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
    1 reference
    static readonly Azure.AI.OpenAI.AzureOpenAIClient client = new Azure.AI.OpenAI.AzureOpenAIClient(new Uri(openAIEndpoint), new Azure.AzureKeyCredential(openAIAPIKey));

    [Microsoft.AspNetCore.Mvc.HttpPost()]
    0 references
    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);
    }
}
```

背後で Azure OpenAI を呼ぶコントローラー

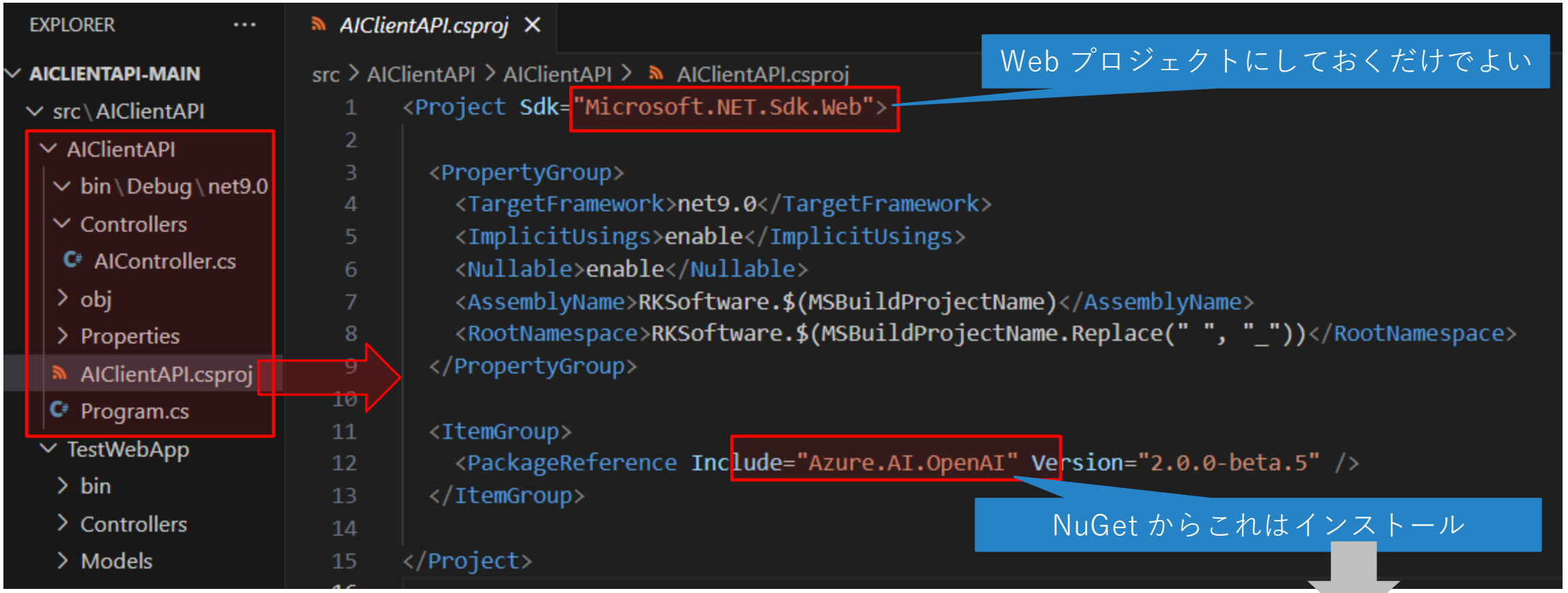
リクエスト・レスポンスデータの型

背後で呼ぶAzure OpenAI の情報
環境変数にしておくことで
開発者ごとの Azure OpenAI が使える(OSSの複数人開発など)
開発・本番での AI の切り替え

なんでもはできないように、
システムプロンプトはサーバーサイドで設定

Azure OpenAI を呼ぶコード

# プロジェクト設定




> dotnet add package Azure.AI.OpenAI

https://www.nuget.org/packages/Azure.AI.OpenAI/

# 追加でこんなコントローラーも

```
[Microsoft.AspNetCore.Mvc.ApiController]
[Microsoft.AspNetCore.Mvc.Route("[controller]")]
0 references
public class AIScriptController : Microsoft.AspNetCore.Mvc.ControllerBase
{
    [Microsoft.AspNetCore.Mvc.HttpGet()]
    0 references
    public string Get() => """
        function addAIClientAPI(idButton, idTextarea, site) {
            document.getElementById(idButton).setAttribute('onclick', `aiClientAPI('${idTextarea}', site);return false;`);
        }
        function aiClientAPI(id, site) {
            alert('要約します');
            var elm = document.getElementById(id);
            fetch(`${site}/AI`, {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ userMessage: elm.value })
            })
            .then(response => response.json())
            .then(data => {
                elm.value = data.aiMessage;
                alert('要約しました');
            })
            .catch(error => {
                alert(error);
            });
        }
    """;
}
```

Web ページ上のボタンをクリックすると
指定のテキストエリアの値を
前述の AI 機能コントローラーに送信する JavaScript

# こんなコードが必要でした

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
var mvcBuilder = builder.Services.AddControllersWithViews();

// dll 読み込み
{
    var basePath = Path.Combine(Directory.GetParent(Assembly.GetEntryAssembly()!.Location!)!.Parent!.Parent!.Parent!.Parent!.FullName, "AIClientAPI\\bin");
    var files = Directory.GetFiles(basePath, "*.dll", SearchOption.AllDirectories).Select(dll => Assembly.LoadFrom(dll)).ToArray();
    foreach (var assembly in files)
    {
        mvcBuilder.AddApplicationPart(assembly);
    }
}
```

DLL 読み込み

コントローラーを追加
（コントローラーを探す場所に DLL を追加）

MvcCoreMvcBuilderExtensions.AddApplicationPart(IMvcBuilder, Assembly) メソッド

## 定義

名前空間: Microsoft.Extensions.DependencyInjection
アセンブリ: Microsoft.AspNetCore.Mvc.Core.dll
パッケージ: Microsoft.AspNetCore.App.Ref v9.0.0

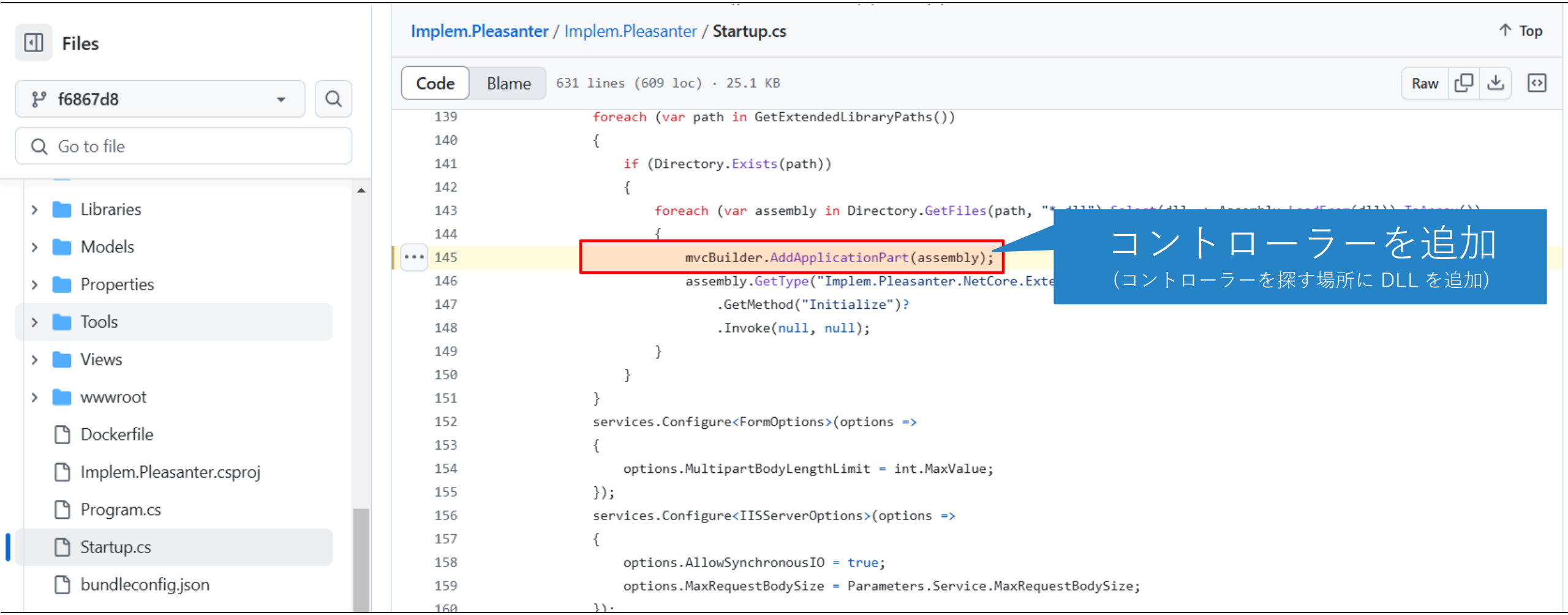
の ApplicationPart 一覧 ApplicationParts に を追加します PartManager。

さらには前述の JavaScript もどうやって呼ぶか……

https://learn.microsoft.com/ja-jp/dotnet/api/microsoft.extensions.dependencyinjection.mvccoremvcbuilderextensions.addapplicationpart

目次

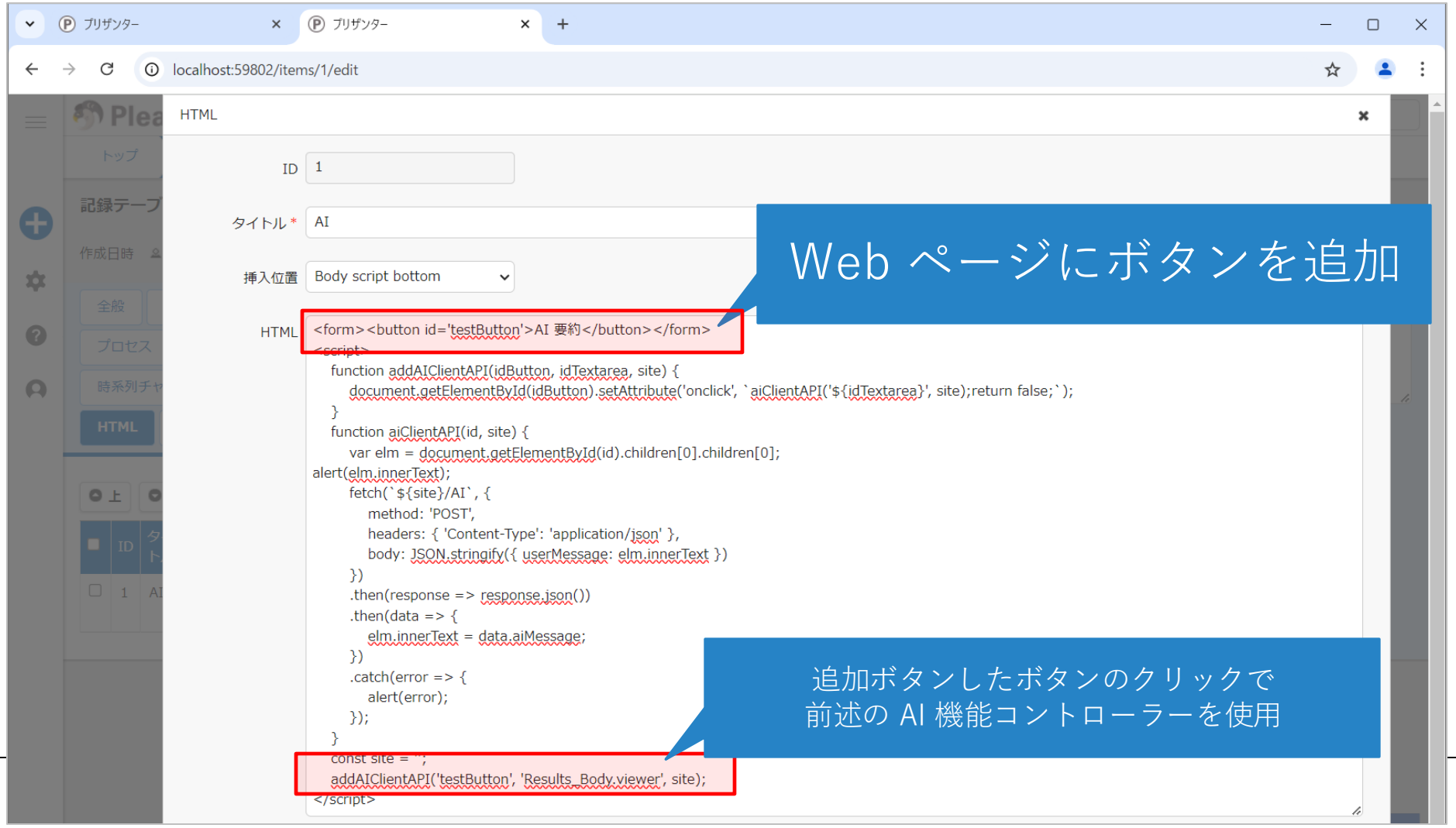
# Pleasanter というプロダクト




https://github.com/Implem/Implem.Pleasanter/blob/f6867d805850d8b912743b346923e3f272de2ea9/Implem.Pleasanter/Startup.cs#L145

# テーブルの管理

**Pleasanter のテーブルの管理という機能は、ページに HTML をコンテンツとして登録可能**



# 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

- C# Tokyo コミュニティではいつでも【登壇者募集中】
- ASP.NET Core MVC では機能追加の口が必要
- 口の用意されているプロジェクトも世の中にはある

# ありがとうございました。

石崎 充良