

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】

The screenshot shows the connpass website interface. At the top, there is a search bar with the text "イベント検索" and a magnifying glass icon, followed by "新着イベント". Below this, there are two yellow notification banners. The first banner, titled "新機能", mentions updates to the connpass API. The second banner, titled "お知らせ", states that as of September 1, 2024, scraping is prohibited. Below the banners, there is a "B! 0" button and a "X ポスト" button. The main content area features a date selector for "12月 9" and a large event title: "【登壇者募集中】 .NET Conf 2024 後! C# Tokyo カンファレンス". The organizer is listed as "主催: C# Tokyo". A large blue hexagonal logo is displayed at the bottom of the event page.

The screenshot shows the YouTube channel page for "C# Tokyo". The channel name "C# Tokyo" is prominently displayed at the top, along with the handle "@CTokyo-wj8fv", "38人" subscribers, and "79本" videos. A "チャンネル登録" button is visible. Below the channel name, there are navigation options: "動画", "ショート", "ライブ", and "再生リスト". The main content area features several video thumbnails. One thumbnail is titled ".NET 9 RC 2!" and another ".NET 9 RC 1!". A central blue banner with the text "【登壇者募集中】" is overlaid on the thumbnails. Below the thumbnails, there is a statistics box for "C# Tokyo" showing "イベント数 117回" and "メンバー数 1747". At the bottom, there is a date and time for an event: "開催前 2024/12/09(月) 19:00 ~ 21:00", with links to "Googleカレンダー" and "icsファイル".

1. **(分量：04 ページ) 人はなぜブログを書くのか**
 2. (分量：03 ページ) 職場ブログを書いてほしい時は
 3. (分量：04 ページ) 自分が良ければ良い
 4. (分量：02 ページ) 締め切り駆動 情報発信

 5. (分量：03 ページ) アプリに何もしないで機能を追加する
 6. (分量：03 ページ) AI 機能の実装は簡単
 7. (分量：03 ページ) ネタバレ・実際に追加してみる

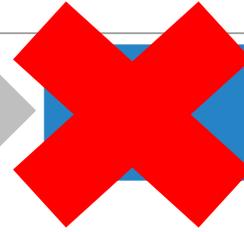
 8. (分量：01 ページ) まとめ
-

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>

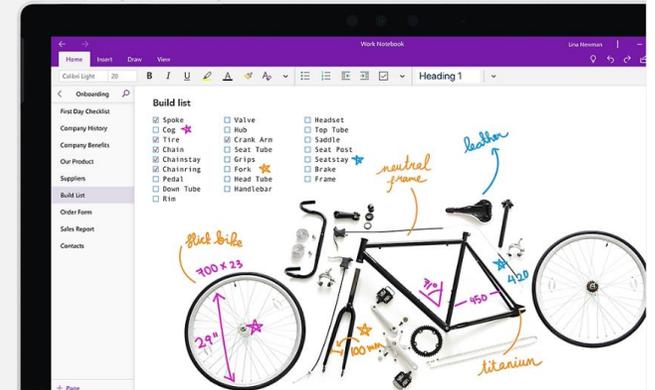
Microsoft OneNote

あなたのデジタル ノートブック

ノート記録が必要なあらゆる場面にこれ1つで対応できる、機能横断型ノートブックです。

サインイン

サインアップ



しかし大きな課題も

おうち vs 職場

学ぶのは自分の環境

個人のアカウント (環境)



OneNote

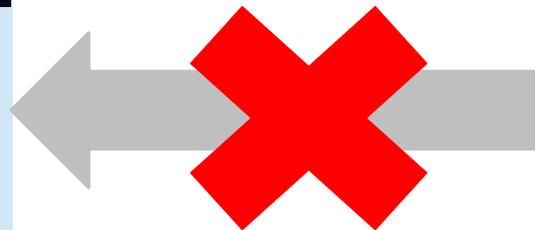
活用は職場

職場の環境



 仕事 PC

参照 (コピペ)
できない



いつでも、どこからでも

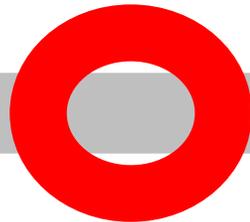
学ぶのは自分の環境

インターネットに公開



ブログ

参照（コピー）
できる



活用は職場

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



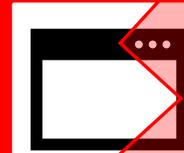
ブログ

ずっと
メンテ可能

職場

職場の環境

メンテ不可



ブログ

転

職場の環境



ブログ

1. (分量：04 ページ) 人はなぜブログを書くのか
 2. **(分量：03 ページ) 職場ブログを書いてほしい時は**
 3. (分量：04 ページ) 自分が良ければ良い
 4. (分量：02 ページ) 締め切り駆動 情報発信

 5. (分量：03 ページ) アプリに何もしないで機能を追加する
 6. (分量：03 ページ) AI 機能の実装は簡単
 7. (分量：03 ページ) ネタバレ・実際に追加してみる

 8. (分量：01 ページ) まとめ
-

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

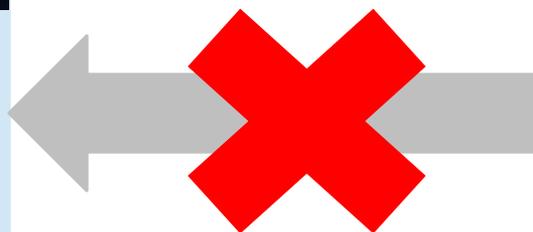
学ぶのは職場

職場の環境



仕事 PC

職場での学びが
ファストだと
自分ブログに書
かない



時系列

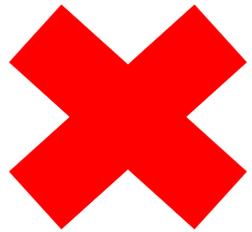
新しいものが登場



調査



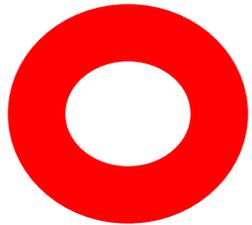
ブログ



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

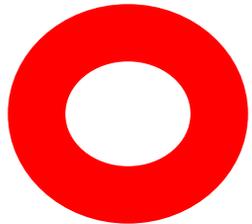
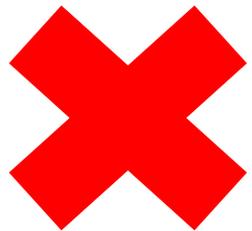


ブログ

調査



ブログ



1. (分量：04 ページ) 人はなぜブログを書くのか
 2. (分量：03 ページ) 職場ブログを書いてほしい時は
 3. **(分量：04 ページ) 自分が良ければ良い**
 4. (分量：02 ページ) 締め切り駆動 情報発信

 5. (分量：03 ページ) アプリに何もしないで機能を追加する
 6. (分量：03 ページ) AI 機能の実装は簡単
 7. (分量：03 ページ) ネタバレ・実際に追加してみる

 8. (分量：01 ページ) まとめ
-

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

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# で早いコードを書きたいけど、ベンチマークの取り方どうやるんだっけ？

benchmark rksoftware

すべて ショッピング 動画 画像 ニュース 地図 書籍 : もっと見る ツー

はてなブログ

<https://rksoftware.hatenablog.com/entry/2023/12/19/> :

メソッドの引数の型違いでパフォーマンス計測 - rksoftware

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) を使用してしまし ...

はてなブログ

<https://rksoftware.hatenablog.com/entry/2023/12/20/> :

キャストと as と TryParse - rksoftware - はてなブログ

2023/12/20 — 最近計測に凝っているので、キャストと as と TryParse も計測してみました。キャストが一番早いのは当然として、as と TryParse が条件によって違う ...

GitHub

[https://github.com/dotnet/Benchm...](https://github.com/dotnet/BenchmarkDotNet) · このページを訳す :

[dotnet/BenchmarkDotNet](https://github.com/dotnet/BenchmarkDotNet): Powerful .NET library for

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();
}
```

例えば

C# で早いコードを
マークの取り方どう

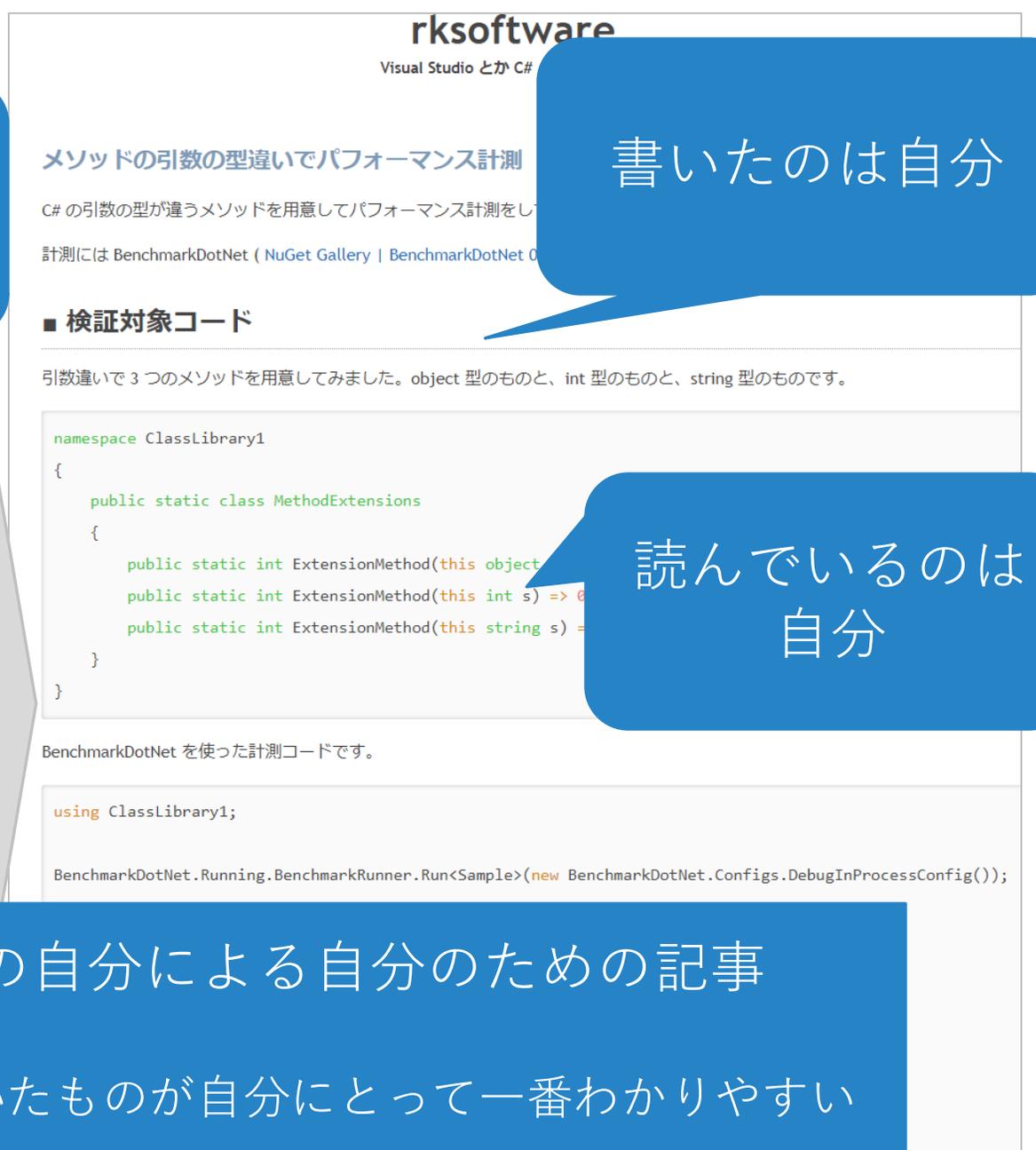
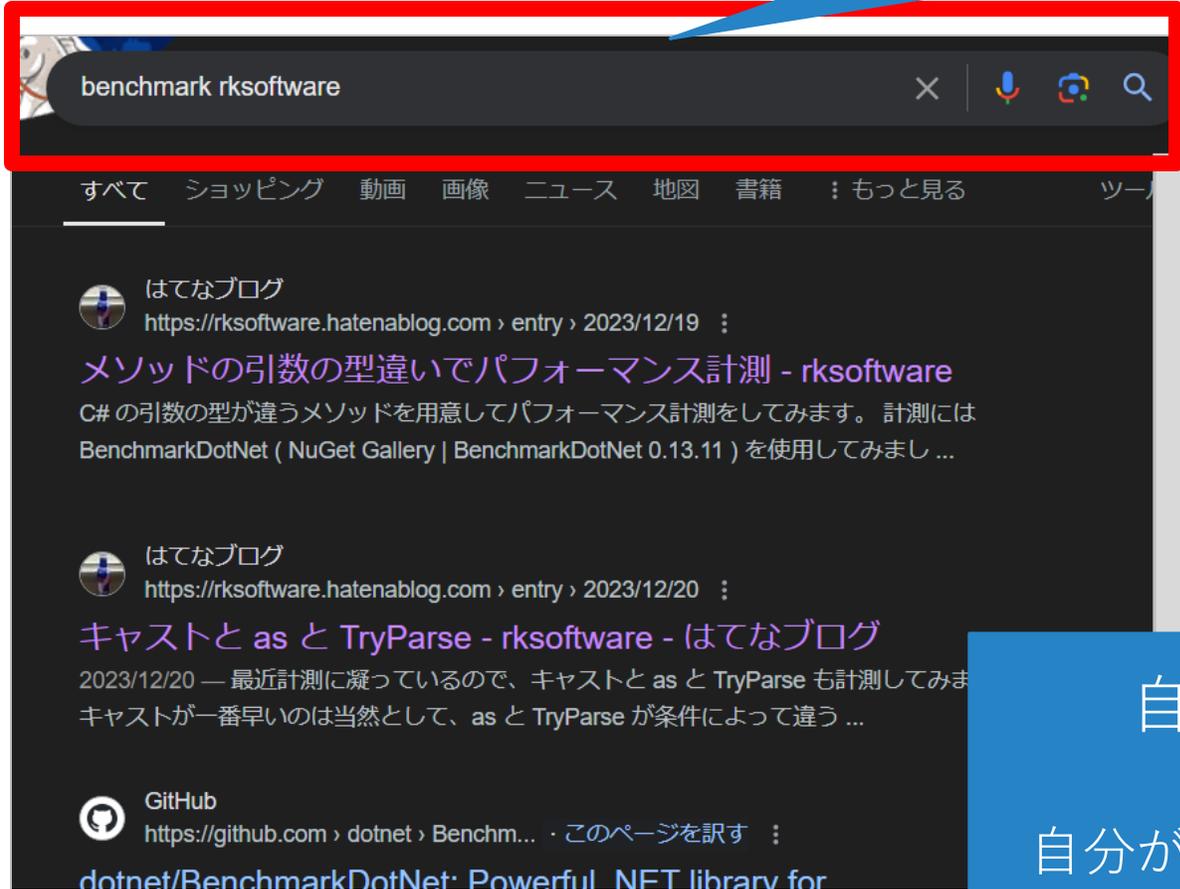
検索したのは自分

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



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

別の切り口・視点・
表現の記事

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

1. (分量 : 04 ページ) 人はなぜブログを書くのか
 2. (分量 : 03 ページ) 職場ブログを書いてほしい時は
 3. (分量 : 04 ページ) 自分が良ければ良い
 4. **(分量 : 02 ページ) 締め切り駆動 情報発信**
 5. (分量 : 03 ページ) アプリに何もしないで機能を追加する
 6. (分量 : 03 ページ) AI 機能の実装は簡単
 7. (分量 : 03 ページ) ネタバレ・実際に追加してみる
 8. (分量 : 01 ページ) まとめ
-

目次



締め切り駆動
情報発信

締め切り駆動 情報発信

connpass produced by Re:muse

イベント検索 🔍 新着イベント ログイン・新規登録

新機能 connpass APIに...
お知らせ 2024年9月1日...
BI 0 📄 ポスト

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

主催: C# Tokyo

C# Tokyo

イベント数 117回
メンバー数 1747

開催前

2024/12/09(月)

18:00

動画公開

C# Tokyo @CTokyo-wj8fv · チャンネル登録者数 38人 · 79本の動画

このチャンネルは...
チャンネル登録

動画 ショート ライブ 再生リスト

YouTube

.NET 9 RC 2! [配信] イベント C# Tokyo 1:04:10

.NET Aspire Day 2024 後! [配信] イベント C# Tokyo 1:04:10

C# Tokyo サイネージ 202411 登録の🔥 YouTube チャンネル... プログラミング言語 C# ユーザーコミュニティ 0:21

.NET 9 RC 1! [配信] イベント C# Tokyo 1:04:10

【配信】NET 9 RC 2! C# Tokyo イベント 15 回視聴・2 週間前

【配信】.NET Aspire Day 2024 後! C# Tokyo イベント 5 回視聴・3 週間前

C# Tokyo サイネージ 202411 15 回視聴・1 か月前

【配信】NET9 RC 1! C# Tokyo イベント 12 回視聴・1 か月前



スライド公開

スライド共有ならDocswell! ログイン 新規登録

人気 <> プログラミング 📁 ビジネス 🏠 教育 🏠 ノウハウ 🏠 科学・技術 🔍 検索

mishizaki @mishizaki mishizaki

このユーザーのスライド一覧 🔍 検索

ソースオープンのススメ

We ❤️ .NET

WinForms の暗黒モードと可変レイアウト

ソースオープンのススメ

.NET の日 ~.NET コミュニティ合同イベント

WinForms の暗黒モードと可変レイアウト

閲覧可能

【登壇者募集中】

connpass produced by Be PROUD

イベント検索 新着イベント

新機能 connpass APIに新しい機能を追加しました。「イベント資料一覧API」「グループ一覧API」を新たに追加し、「イベント申し込み」しました。詳細な仕様や利用方法は、[APIリファレンス](#)をご確認ください。API利用を希望される方は、[connpassのAPI利用について](#)

お知らせ 2024年9月1日より、connpassではスクレイピングを禁止し、利用規約に明記しました。以降の情報取得にはconnpassをご確認ください。

B! 0 投稿

12月9日 【登壇者募集中】 .NET Conf 2024 後! C# Tokyo Conference

主催: C# Tokyo

C# Tokyo @CTokyo-wj8fv · チャンネル登録者数 38人 · 79本の動画

このチャンネルの詳細...さらに表示

チャンネル登録

動画 ショート ライブ 再生リスト

.NET 9 RC 2! 【配信】イベント

.NET Aspire Day 2024 後! 【配信】イベント

C# Tokyo プログラミング言語 C# ユーザーコミュニティ 0:21

.NET 9 RC 1! 【配信】イベント C# Tokyo 1:04:10

サイネージ 202411 : 【配信】NET9RC 1! C#Tokyoイベント

1 か月前 12 回視聴 · 1 か月前

C# Tokyo

イベント数 117回

メンバー数 1747

開催前

2024/12/09(月)

19:00 ~ 21:00

Googleカレンダー icsファイル

【登壇者募集中】



アプリに何もしないで
アプリに AI を組み込む

1. (分量：04 ページ) 人はなぜブログを書くのか
 2. (分量：03 ページ) 職場ブログを書いてほしい時は
 3. (分量：04 ページ) 自分が良ければ良い
 4. (分量：02 ページ) 締め切り駆動 情報発信
 5. **(分量：03 ページ) アプリに何もしないで機能を追加する**
 6. (分量：03 ページ) AI 機能の実装は簡単
 7. (分量：03 ページ) ネタバレ・実際に追加してみる
 8. (分量：01 ページ) まとめ
-

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](#)。

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

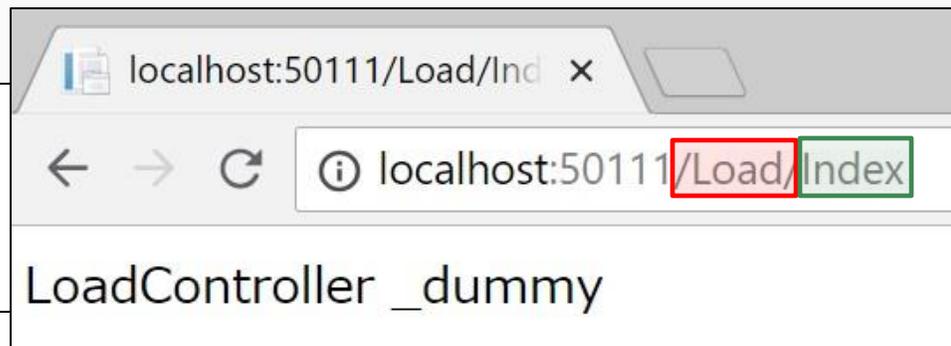
```
routes.MapRoute(  
    name: "Default",  
    url: "{controller}/{action}/{id}",  
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }  
);
```

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

The screenshot shows the Visual Studio interface with the following elements:

- Project Explorer (Left):** Shows a solution named 'AICLIENAPI-MAIN'. Underneath, there is a folder 'src\AIClientAPI'. Inside this folder, there is a sub-folder 'AIClientAPI' which contains 'bin', 'Controllers', 'obj', 'Properties', 'AIClientAPI.csproj', and 'Program.cs'. Below this is another folder 'TestWebApp' containing 'bin', 'Controllers', 'Models', 'obj', 'Properties', 'Views', 'wwwroot', 'AIClientAPI.http', 'appsettings.Development...', 'appsettings.json', and 'Program.cs'.
- Code Editor (Right):** Shows the 'Program.cs' file. The code includes using statements for 'Microsoft.Extensions.DependencyInjection' and 'System.Reflection'. It defines a 'builder' and an 'mvcBuilder'. A comment indicates 'dll 読み込み' (DLL loading). The code uses a long relative path to load DLLs from the 'bin' directory: `Directory.GetParent(Assembly.GetEntryAssembly()!.Location)!.Parent!.Parent!.Parent!.Parent!.FullName, "AIClientAPI\\bin")`. It then iterates over the loaded assemblies and adds them to the application.

存在感の強い長さの相対パス

ASP.NET Core MVC プロジェクト

1. (分量：04 ページ) 人はなぜブログを書くのか
 2. (分量：03 ページ) 職場ブログを書いてほしい時は
 3. (分量：04 ページ) 自分が良ければ良い
 4. (分量：02 ページ) 締め切り駆動 情報発信

 5. (分量：03 ページ) アプリに何もしないで機能を追加する
 6. **(分量：03 ページ) AI 機能の実装は簡単**
 7. (分量：03 ページ) ネタバレ・実際に追加してみる

 8. (分量：01 ページ) まとめ
-

目次

AI 機能を実装したコントローラー

背後で Azure OpenAI を呼ぶコントローラー

```
namespace RKSoftware.AIClientAPI.Controllers;  
  
[Microsoft.AspNetCore.Mvc.ApiController]  
[Microsoft.AspNetCore.Mvc.Route("[controller]")]  
0 references  
public class AIController : Microsoft.AspNetCore.Mvc.ControllerBase  
{
```

リクエスト・レスポンスデータの型

```
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、コピー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")!;
```

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

```
// 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
```

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

```
static readonly Azure.AI.OpenAI.AzureOpenAIClient client = new Azure.AI.OpenAI.AzureOpenAIClient(new Uri(openAIEndpoint), new Azure.AzureKeyCredential(openAIAPIKey));
```

Azure OpenAI を呼ぶコード

```
[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);
```

プロジェクト設定

Web プロジェクトにしておくだけでよい

```
1 <Project Sdk="Microsoft.NET.Sdk.Web">
2
3   <PropertyGroup>
4     <TargetFramework>net9.0</TargetFramework>
5     <ImplicitUsings>enable</ImplicitUsings>
6     <Nullable>enable</Nullable>
7     <AssemblyName>RKSoftware.$(MSBuildProjectName)</AssemblyName>
8     <RootNamespace>RKSoftware.$(MSBuildProjectName.Replace(" ", "_"))</RootNamespace>
9   </PropertyGroup>
10
11   <ItemGroup>
12     <PackageReference Include="Azure.AI.OpenAI" Version="2.0.0-beta.5" />
13   </ItemGroup>
14
15 </Project>
```

NuGet からこれはインストール

> dotnet add package 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……

1. (分量 : 04 ページ) 人はなぜブログを書くのか
 2. (分量 : 03 ページ) 職場ブログを書いてほしい時は
 3. (分量 : 04 ページ) 自分が良ければ良い
 4. (分量 : 02 ページ) 締め切り駆動 情報発信

 5. (分量 : 03 ページ) アプリに何もしないで機能を追加する
 6. (分量 : 03 ページ) AI 機能の実装は簡単
 7. **(分量 : 03 ページ) ネタバレ・実際に追加してみる**

 8. (分量 : 01 ページ) まとめ
-

目次

Pleasanter というプロダクト

Files

f6867d8

Go to file

- Libraries
- Models
- Properties
- Tools
- Views
- wwwroot
- Dockerfile
- Implem.Pleasanter.csproj
- Program.cs
- Startup.cs
- bundleconfig.json

Implem.Pleasanter / Implem.Pleasanter / Startup.cs

Code Blame 631 lines (609 loc) · 25.1 KB

Raw Copy Download

```
139     foreach (var path in GetExtendedLibraryPaths())
140     {
141         if (Directory.Exists(path))
142         {
143             foreach (var assembly in Directory.GetFiles(path, "*.dll"))
144             {
145                 mvcBuilder.AddApplicationPart(assembly);
146                 assembly.GetType("Implem.Pleasanter.NetCore.Extensions")
147                     .GetMethod("Initialize")?
148                     .Invoke(null, null);
149             }
150         }
151     }
152     services.Configure<FormOptions>(options =>
153     {
154         options.MultipartBodyLengthLimit = int.MaxValue;
155     });
156     services.Configure<IISServerOptions>(options =>
157     {
158         options.AllowSynchronousIO = true;
159         options.MaxRequestBodySize = Parameters.Service.MaxRequestBodySize;
160     });
```

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)

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

Web ページにボタンを追加

```
HTML <form><button id='testButton'>AI 要約</button></form>
<script>
function addAIClientAPI(idButton, idTextarea, site) {
  document.getElementById(idButton).setAttribute('onclick', `aiClientAPI('${idTextarea}', site);return false;`);
}
function aiClientAPI(id, site) {
  var elm = document.getElementById(id).children[0].children[0];
  alert(elm.innerText);
  fetch(`${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>
```

追加ボタンしたボタンのクリックで
前述の AI 機能コントローラーを使用

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：04 ページ) 人はなぜブログを書くのか
 2. (分量：03 ページ) 職場ブログを書いてほしい時は
 3. (分量：04 ページ) 自分が良ければ良い
 4. (分量：02 ページ) 締め切り駆動 情報発信

 5. (分量：03 ページ) アプリに何もしないで機能を追加する
 6. (分量：03 ページ) AI 機能の実装は簡単
 7. (分量：03 ページ) ネタバレ・実際に追加してみる

 8. **(分量：01 ページ) まとめ**
-

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