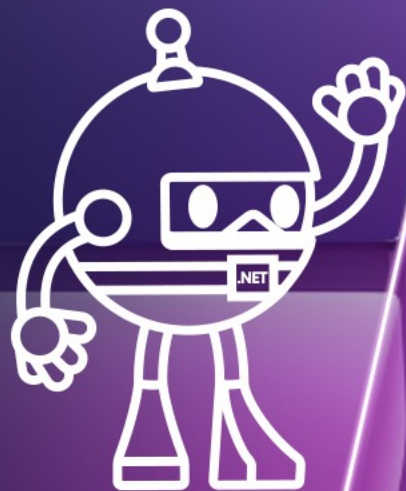


.NET Conf China 2023

2023/12/16
09:30 - 18:00

中国 · 北京



.NET应用国际化-AIGC智能翻译+代码生成

周国庆 2023/12/16



周国庆

特来电云平台技术负责人

.NET领域-微软MVP (2020-2022)

济南.NET俱乐部主席



今天和大家分享的内容：

随着疫情的消退，越来越多的企业开始向海外拓展，应用系统的国际化和本地化是一个巨大的技术挑战，我们今天重点探讨以下内容：

- .NET应用如何实现国际化？不仅仅包含资源文件和文本的替换，还有文本词条抽取、智能翻译、代码替换、本地化处理等各种场景。
- 基于Roslyn进行代码分析，查找中文文本、抽取词条，以及代码替换。
- 机器翻译与GPT的Battle，基于GPT4实现一个智能翻译服务。
- 彩蛋环节：如何使用Github Copilot自动生成代码和单元测试。



一、业务背景

特来电新能源股份有限公司，中国最大最强新能源汽车充电网运营商，主要从事新能源汽车充电网的建设、运营以及互联网增值服务。



我们已经	亿	万	万	万	亿	万
做到	25,198,856,918		476,447	360	25,758,832	11,199,491,964
	总充电量 (度)		运营终端数	覆盖城市	碳减排 (吨)	节油量 (升)

充电桩



47w+

日充电量



3200w+

注册用户



2000w+

日均服务调用



100亿+

日均数据存储



10T+

日均消息处理



10亿+

2022年，特来电正式启动出海业务，产品和应用全面支持国际化，走向海外。



二、.NET应用如何实现国际化

目前我们的充电服务平台包含16大子系统，上千个功能菜单，数十个数据库...

大部分应用基于.NET技术栈构建，都需要支持国际化&本地化。

如何快速、高效、准确地完成产品国际化&本地化改造是团队面临的一个巨大的挑战!

没有相关经验怎么办?

好在有了ChatGPT!!!

我们先问一下ChatGPT~



二、.NET应用如何实现国际化

GPT4 果然牛



.NET应用如何实现国际化?



.NET 应用程序的国际化（也称为本地化）涉及将应用程序设计成可以适应不同的语言和区域设置，而不需要重新编写代码。这通常包括翻译文本、支持不同的日期和时间格式以及处理不同的货币单位等。以下是实现 .NET 应用程序国际化的几个关键步骤：

- 资源文件：**创建资源文件（通常是 .resx 文件），用于存储可本地化的字符串和其他文化相关的资源。每种语言/区域设置都应有其相应的资源文件，例如 `Messages.en.resx` (英文), `Messages.fr.resx` (法文) 等。
- 文化信息：**使用 `System.Globalization.CultureInfo` 类来设置和获取当前线程的文化信息。这会影响日期、时间、数字和货币的格式。
- 资源管理器：**使用 `ResourceManager` 类来检索适当文化的资源。这个类会根据当前文化自动查找和加载正确的资源文件。
- 字符串外部化：**确保所有用户界面元素和需要显示给用户的字符串都存储在资源文件中，而不是硬编码在应用程序中。
- 日期和数字格式化：**使用 `DateTime.ToString()`、`double.ToString()` 等方法来格式化日期、时间和数字，这些方法会根据当前文化自动应用适当的格式。
- 文本方向：**对于从右向左（RTL）阅读的语言（如阿拉伯语和希伯来语），确保用户界面可以适当地调整。
- 测试：**在开发过程中进行彻底的国际化测试，以确保应用程序在不同的文化设置中表现正确。
- 第三方库：**在需要时使用第三方库来处理复杂的国际化需求，如高级日期处理、复数形式和文化特定的校验。

二、.NET应用如何实现国际化

以下是一个简单的示例，展示如何在 .NET 应用程序中使用资源文件来国际化：

1. 创建资源文件：

- `Strings.resx`：默认资源（通常为英文）
- `Strings.fr.resx`：法文资源

2. 在资源文件中添加键值对，例如：

- `WelcomeText` 对应 "Welcome" 在 `Strings.resx` 和 "Bienvenue" 在 `Strings.fr.resx`。

3. 在代码中使用资源：

```
using System;
using System.Globalization;
using System.Resources;
using System.Threading;

class Program
{
    static void Main()
    {
        // 设置当前线程的文化为法语
        Thread.CurrentThread.CurrentUICulture = new CultureInfo("fr-FR");

        // 创建 ResourceManager 实例
        ResourceManager rm = new ResourceManager("YourAppNamespace.Strings", typeof(Program).Assembly);

        // 获取适当文化的字符串
        string welcomeText = rm.GetString("WelcomeText");

        Console.WriteLine(welcomeText); // 输出 "Bienvenue"
    }
}
```

在开发过程中，应始终考虑国际化，这样可以更容易地在之后添加新的语言支持。记住，国际化不仅仅是翻译文本，还包括考虑文化差异和各种格式化需求。

技术方案可行，但是有几个架构设计上的问题：

1. 重度依赖资源文件

2. 大型分布式部署，分发、管理成本很高、很复杂

3. 需要大范围扫描、改造代码

4. 翻译、校对工作量巨大

5. 无法批量、动态修改翻译文本



二、.NET应用如何实现国际化

在ChatGPT基础上，发挥.NET技术栈的能力，创新设计了一个新的解决方案：

- 抽象封装一个词条服务，根据线程上下文的CultureInfo，动态获取对应的多语言文本

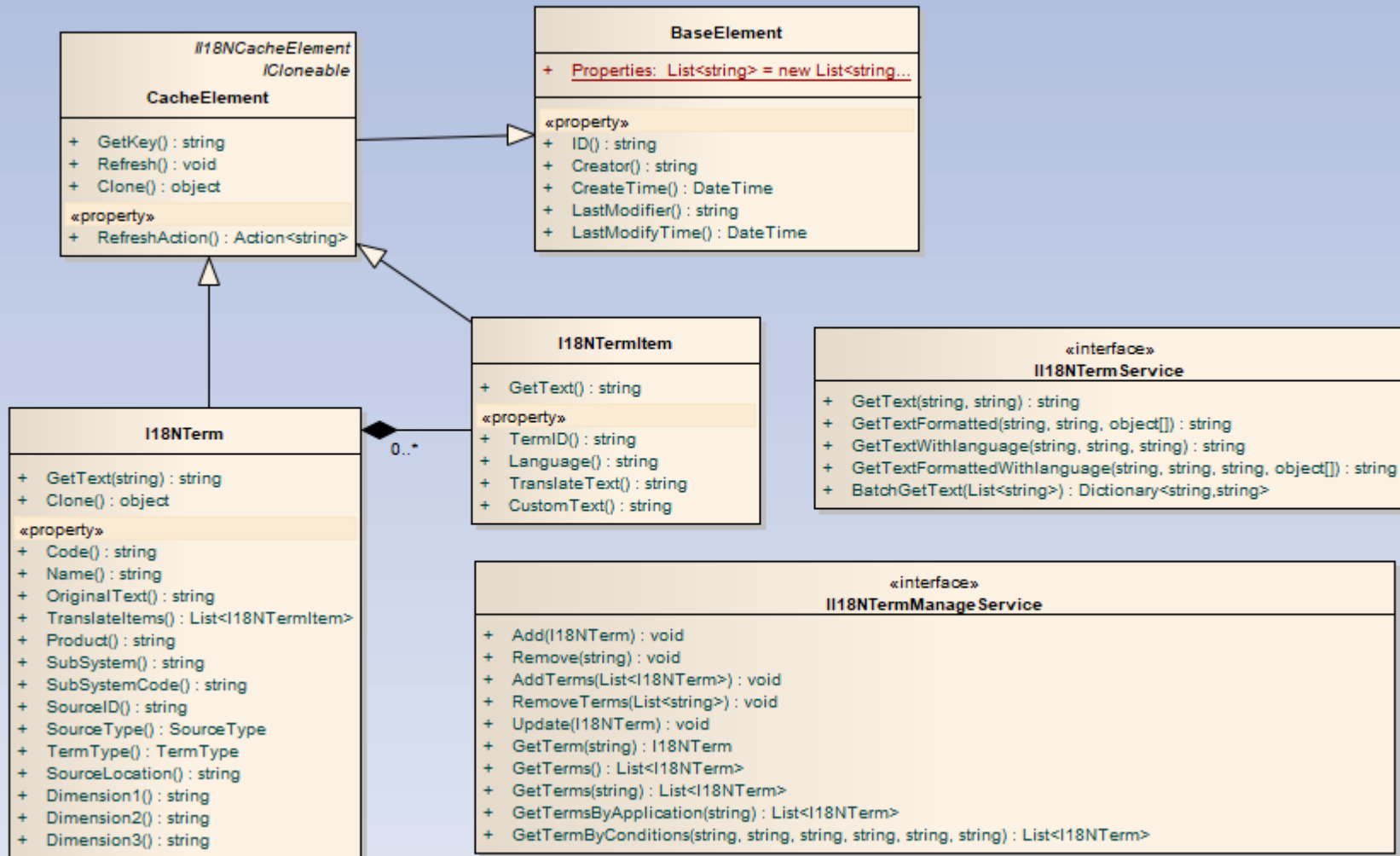
注：词条通常用于标识需要被翻译文本的唯一标识。

- 基于Roslyn，解析代码中的中文，形成多语言词条，同时做代码替换
- 封装一个翻译服务，批量翻译词条
- 统一管理所有的多语言词条，支持动态修改
- 提供本地化的SDK，实现时间、数字、单位的本地化



二、.NET应用如何实现国际化

抽象封装词条和词条服务



词条类I18NTerm: 用于存储词条数据

词条管理接口: 用于词条的批量新增、修改

词条服务接口: 支持按词条查询对应的翻译文本。



二、.NET应用如何实现国际化

基于Roslyn，解析识别代码中的中文，批量进行代码替换

再次找到ChatGPT问一问

Prompt: 你是一个.NET资深开发工程师，全面掌握C#语言，请基于Roslyn技术实现一个服务，输入一个sln解决方案的路径，扫描各个类中的中文文字，统一替换为I18nTermService.GetText('词条ID')



你是一个.NET资深开发工程师，全面掌握C#语言，请基于Roslyn技术实现一个服务，输入一个sln解决方案的路径，扫描各个类中的中文文字，统一替换为I18nTermService.GetText('词条ID')

要实现这样一个服务，您需要以下步骤：

1. 解析sln文件，找出所有的csproj项目文件。
2. 遍历每个csproj文件，找到其中的C#源文件(.cs)。
3. 使用Roslyn编译器来遍历每个C#源文件中的语法树(Syntax Tree)。
4. 识别出所有包含中文文本的字符串字面量。
5. 将每个中文文本的字符串字面量替换为I18nTermService.GetText('词条ID')的调用。
6. 生成一个词条ID与中文文本的映射，以便于后续的国际化工作。
7. 将修改后的代码写回到源文件。

下面提供一个简单的代码示例，演示如何使用Roslyn分析和替换代码中的中文字符串。请注意，这个示例并不包括解析sln文件和遍历所有csproj文件的过程，而是直接针对一个C#源文件进行操作。

首先，确保您安装了必要的NuGet包：

```
Install-Package Microsoft.CodeAnalysis.CSharp
Install-Package Microsoft.CodeAnalysis.CSharp.Workspaces
```

然后，您可以使用以下C#代码作为服务的基础：



二、.NET应用如何实现国际化

基于Roslyn，解析识别代码中的中文，批量进行代码替换

```
class Program
{
    static async System.Threading.Tasks.Task Main(string[] args)
    {
        string solutionPath = args[0]; // 从命令行参数获取解决方案路径
        MSBuildWorkspace workspace = MSBuildWorkspace.Create();
        Solution solution = await workspace.OpenSolutionAsync(solutionPath);

        foreach (var projectId in solution.ProjectIds)
        {
            Project project = solution.GetProject(projectId);

            foreach (Document document in project.Documents)
            {
                SyntaxNode syntaxRoot = await document.GetSyntaxRootAsync();
                var literalExpressions = syntaxRoot.DescendantNodes()
                    .OfType<LiteralExpressionSyntax>()
                    .Where(l => l.IsKind(SyntaxKind.StringLiteralExpression) && IsChineseText(l.Token.ValueText));

                var newSyntaxRoot = syntaxRoot;
                foreach (var literalExpression in literalExpressions)
                {
                    string text = literalExpression.Token.ValueText;
                    string termId = GenerateTermId(text); // 生成词条ID的逻辑
                    var newLiteralExpression = SyntaxFactory.ParseExpression($"I18nTermService.GetText(\"{termId}\")")
                        .WithTriviaFrom(literalExpression);

                    newSyntaxRoot = newSyntaxRoot.ReplaceNode(literalExpression, newLiteralExpression);
                }

                if (newSyntaxRoot != syntaxRoot)
                {
                    Document newDocument = document.WithSyntaxRoot(newSyntaxRoot);
                    SyntaxNode formattedNode = Formatter.Format(newSyntaxRoot, workspace);
                    File.WriteAllText(document.FilePath, formattedNode.ToFullString());
                }
            }
        }
    }
}
```

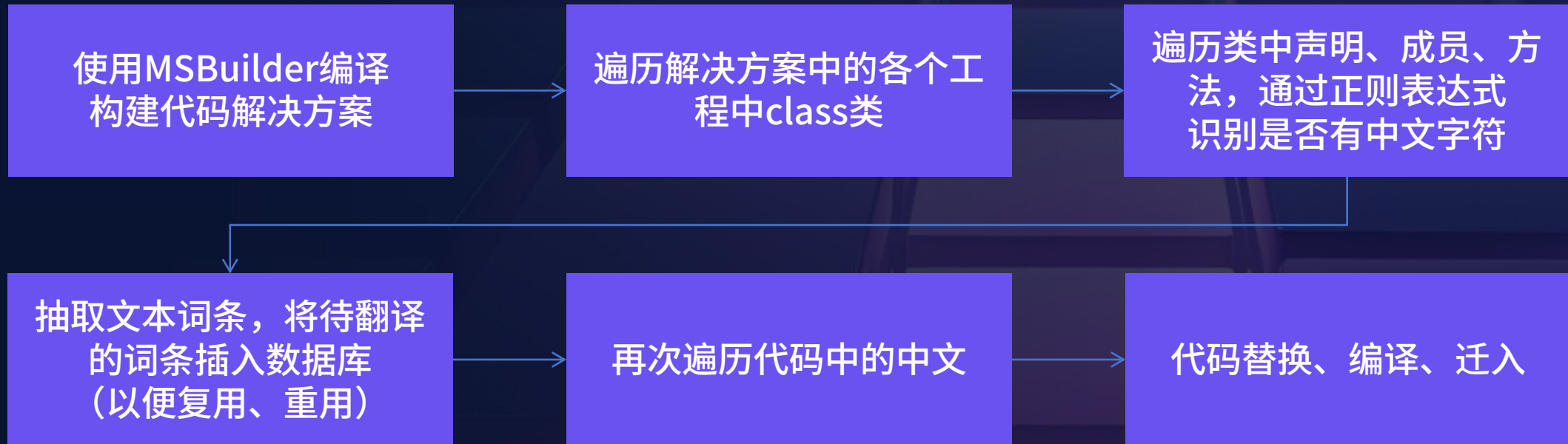


ChatGPT给出了一个可行的技术方案



二、.NET应用如何实现国际化

基于Roslyn，解析识别代码中的中文，批量进行代码替换



```
throw new TeldException(OCMExceptionCode.GetAdjustChargOrdersFail, TermService.Current.GetText("T_Standard_OCM_00000183", "获取调整单列表失败"), ex, TeldExceptionLevel.Warning);
```



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

基于Azure AI services 的 Translator实现机器翻译

```
private string Translate(string text, string language)
{
    using (var client = new HttpClient())
    {
        using (var request = new HttpRequestMessage())
        {
            var url = $"/translate?api-version=3.0&to={language}";
            request.Method = HttpMethod.Post;
            request.RequestUri = new Uri($"{_endpoint}{url}");
            object[] body = { new { Text = text } };
            var requestBody = JsonConvert.SerializeObject(body);
            request.Content = new StringContent(requestBody, Encoding.UTF8, "application/json");
            request.Headers.Add("Ocp-Apim-Subscription-Key", _key);

            var response = client.SendAsync(request).ConfigureAwait(false).GetAwaiter().GetResult();
            //TODO 可以对状态码做处理
            if (!response.IsSuccessStatusCode)
            {
                return null;
            }
            string result = response.Content.ReadAsStringAsync().Result;
            var translationResults = JsonConvert.DeserializeObject<TranslationResult[]>(result);
            if (translationResults.Length > 0)
            {
                return translationResults[0].Translations.FirstOrDefault()?.Text;
            }
        }
    }
    return null;
}
```



大规模-批量翻译词条



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

机器翻译的准确性？ 机器翻译的问题

中文	英文
充电站	Charging stations
电站	power station
充电桩	Charging piles
充电终端	Charging terminals
终端	terminal
...	



缺少语义上下文
缺少专业领域术语表
导致翻译不准确
翻译出的结果海外用户看不懂





二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

尝试使用ChatGPT做专业翻译：**先设计Prompt**

请把以下词语列表翻译为英文 1. 充电站, 2. 电站, 3. 充电桩, 4. 充电终端, 5. 终端, 每个单词一行

 请把以下词语列表翻译为英文, 1. 充电站, 2. 电站, 3. 充电桩, 4. 充电终端, 5. 终端, 每个单词一行,

-  1. Charging Station
2. Power Station
3. Charging Pile
4. Charging Terminal
5. Terminal

依旧不理想, 继续修改Prompt



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

Prompt: 你是一个美国电动汽车充电服务运营商，精通中文和英文，请使用专业领域术语，把以下词语列表翻译为英文，1. 充电站，2. 电站，3. 充电桩，4. 充电终端，5. 终端，翻译时电站等同于充电站，充电终端等同于充电桩，终端也等同于充电桩，每个单词一行。

ChatGPT

 你是一个美国电动汽车充电服务运营商，精通中文和英文，请使用专业领域术语，把以下词语列表翻译为英文，1. 充电站，2. 电站，3. 充电桩，4. 充电终端，5. 终端，翻译时电站等同于充电站，充电终端等同于充电桩，终端也等同于充电桩，每个单词一行。

 1. Charging Station
2. Charging Station
3. Charging Pile
4. Charging Pile
5. Charging Pile



充电桩的专业翻译是
Charging point
需要一个专业术语表



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

Prompt: 你是一个美国电动汽车充电服务运营商，精通中文和英文，请使用以下格式的专业术语 {"充电站":"Charging station", "充电桩":"Charging point"}，把以下词语列表翻译为英文，1. 充电站，2. 电站，3. 充电桩，4. 充电终端，5. 终端，翻译时电站等同于充电站，充电终端等同于充电桩，终端也等同于充电桩，每个单词一行。

ChatGPT

你是一个美国电动汽车充电服务运营商，精通中文和英文，请使用以下格式的专业术语 {"充电站":"Charging station", "充电桩":"Charging point"}，把以下词语列表翻译为英文，1. 充电站，2. 电站，3. 充电桩，4. 充电终端，5. 终端，翻译时电站等同于充电站，充电终端等同于充电桩，终端也等同于充电桩，每个单词一行。

1. Charging station
2. Charging station
3. Charging point
4. Charging point
5. Charging point



翻译准确性提升了
我们继续改进，同时实现工程化



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

Prompt: 你是一个美国电动汽车充电服务运营商，精通中文和英文，请使用以下格式的专业术语 {"充电站":"Charging station", "充电桩":"Charging point"}，把以下词语列表翻译为英文，1. 充电站，2. 电站，3. 充电桩，4. 充电终端，5. 终端，翻译时电站等同于充电站，充电终端等同于充电桩，终端也等同于充电桩，请以JSON格式返回，例如 {"充电站":"Charging station", "充电终端":"Charging point"}，不需要做解释

ChatGPT



你是一个美国电动汽车充电服务运营商，精通中文和英文，请使用以下格式的专业术语 {"充电站":"Charging station", "充电桩":"Charging point"}，把以下词语列表翻译为英文，1. 充电站，2. 电站，3. 充电桩，4. 充电终端，5. 终端，翻译时电站等同于充电站，充电终端等同于充电桩，终端也等同于充电桩，请以JSON格式返回，例如 {"充电站":"Charging station", "充电终端":"Charging point"}，不需要做解释



```
{  
  "充电站": "Charging station",  
  "电站": "Charging station",  
  "充电桩": "Charging point",  
  "充电终端": "Charging point",  
  "终端": "Charging point"  
}
```



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

更好的Prompt

请扮演一个美国电动汽车充电服务运营商，精通中文和英文，请使用以下专业术语

```
{"充电站":"Charging station", "电站":"Charging station", "场站":"Charging station", "充电桩":"Charging point", "充电终端":"Charging point", "终端":"Charging point", "电动汽车":"Electric Vehicle", "直流快充":"DC Fast Charger", "超级充电站":"Supercharger", "智能充电":"Smart Charging", "交流慢充":"AC Slow Charging"},
```

把请将用户的输入翻译为英文，请以JSON格式返回

```
例如 {"充电站":"Charging station", "充电终端":"Charging point"}
```

不需要做解释



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

ChatGPT

请扮演一个美国电动汽车充电服务运营商，精通中文和英文，请使用以下专业术语 ("充电站":"Charging station", "电站":"Charging station", "场站":"Charging station", "充电桩":"Charging point", "充电终端":"Charging point", "终端":"Charging point", "电动汽车":"Electric Vehicle", "直流快充":"DC Fast Charger", "超级充电站":"Supercharger", "智能充电":"Smart Charging", "交流慢充":"AC Slow Charging"), 把请用户的输入翻译为英文, 请以JSON格式返回, 例如 {"充电站":"Charging station", "充电终端":"Charging point"}, 不需要做解释

Sure, please provide the user's input that you would like me to translate into English.

充电桩

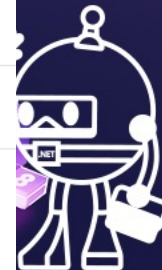
```
{  
  "充电桩": "Charging point"  
}
```

充电站

```
{  
  "充电站": "Charging station"  
}
```

当前电站不可用

```
{  
  "当前电站不可用": "The current Charging station is not available"  
}
```



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

使用SK框架，基于GPT4实现翻译服务, 用于专业翻译



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

```
public class GPT4TranslateService
{
    2 references
    public IDictionary<int,string> Translate(IDictionary<int, string> texts, CultureInfo cultureInfo)
    {
        var kernel = BuildKernel();
        var translator = new Translator(kernel);
        return translator.Translate(texts, cultureInfo.TwoLetterISOLanguageName );
    }

    //私有方法, 构造IKernel
    1 reference
    private IKernel BuildKernel()
    {
        var builder = new KernelBuilder();
        builder.WithAzureOpenAIChatCompletionService(...
        var kernel = builder.Build();
        return kernel;
    }
}
```



二、.NET应用如何实现国际化

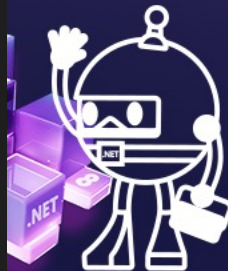
机器翻译VSChatGPT翻译

```
1 reference
public IDictionary<int, string> Translate(IDictionary<int, string> textList, string language)
{
    var pluginDirectory = Path.Combine(System.IO.Directory.GetCurrentDirectory(), "Plugins");
    var plugin = kernel.ImportSemanticFunctionsFromDirectory(pluginDirectory, "TranslatePlugin");

    var json = JsonConvert.SerializeObject(textList);

    if (!string.IsNullOrEmpty(json))
    {
        var output = kernel.RunAsync(json, plugin["Translator_" + language + ""]).Result.ToString();
        if (!string.IsNullOrWhiteSpace(output))
        {
            Console.WriteLine(output);
            return JsonConvert.DeserializeObject<Dictionary<int, string>>(output);
        }
    }

    return new Dictionary<int, string>();
}
```

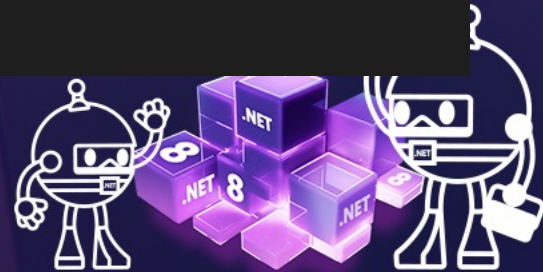


二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

- TranslatePlugin
 - Translator_de
 - Translator_en
 - Translator_fr
 - Translator_jp
 - Translator_kr
 - Translator_pt

```
1  {{{input}}}
2  请将上面的输入翻译为英文,不要返回任何解释说明,
3  请扮演一个美国电动汽车充电服务运营商(精通中文和英文),用户的输入数据是JSON格式,例如{"1":"充电站", "2":"充电桩"},
4  如果不是JSON格式,请返回无效的输入。
5  请使用以下专业术语进行翻译
6  {
7    "充电站":"Charging station",
8    "电站":"Charging station",
9    "场站":"Charging station",
10   "充电桩":"Charging point",
11   "充电终端":"Charging point",
12   "终端":"Charging point",
13   "电动汽车":"Electric Vehicle",
14   "直流快充":"DC Fast Charger",
15   "超级充电站":"Supercharger",
16   "智能充电":"Smart Charging",
17   "交流慢充":"AC Slow Charging"
18 }
19 翻译结果请以JSON格式返回,例如 {"1":"Charging station", "2":"Charging point"}
```



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

- TranslatePlugin
 - Translator_de
 - Translator_en
 - Translator_fr
 - Translator_jp
 - Translator_kr
 - Translator_pt

```
1  {{{input}}}
2  请将上面的输入翻译为葡萄牙语,不要返回任何解释说明,请扮演一个巴西的电动汽车充电服务运营商(精通葡萄牙语、中文和英文)
3  用户的输入数据是JSON格式,例如{"1":"充电站","2":"充电桩"},如果不是JSON格式,请返回无效的输入
4  请使用以下专业术语进行翻译
5  {
6    "充电站": "Estação de carregamento",
7    "电站": "Estação de carregamento",
8    "场站": "Estação de carregamento",
9    "充电桩": "Ponto de carregamento",
10   "充电终端": "Ponto de carregamento",
11   "终端": "Ponto de carregamento",
12   "电动汽车": "Veículo Elétrico",
13   "直流快充": "Carregador Rápido DC",
14   "超级充电站": "Supercharger",
15   "智能充电": "Carregamento Inteligente",
16   "交流慢充": "Carregamento AC Lento"
17 }
18 请以JSON格式返回,例如 {"1":"Estação de carregamento", "2":"Ponto de carregamento"}
```



二、.NET应用如何实现国际化

机器翻译VSChatGPT翻译

```
var culture = new CultureInfo("en-US");
var translator = new GPT4TranslateService();
translator.Translate(new Dictionary<int, string>(){ { 1, "当前电站不可用"}, { 2, "终端不可用"}, { 3, "充电桩不可用"},
{ 4, "代码质量"}, { 5, "数量" } }, culture);

culture = new CultureInfo("pt-BR");
translator.Translate(new Dictionary<int, string>(){ { 1, "当前电站不可用"}, { 2, "终端不可用"}, { 3, "充电桩不可用"},
{ 4, "代码质量"}, { 5, "数量" } }, culture);
```

```
{
  "1": "The current charging station is not available",
  "2": "Charging point is not available",
  "3": "Charging point is not available",
  "4": "Code quality",
  "5": "Quantity"
}
{
  "1": "Esta??o de carregamento atual n?o est? dispon?vel",
  "2": "Ponto de carregamento n?o est? dispon?vel",
  "3": "Ponto de carregamento n?o est? dispon?vel",
  "4": "Qualidade do c?digo",
  "5": "Quantidade"
}
```



JSON结构化的数据
写到词条翻译结果表中
基于这个数据生成产品的多语言包



三、使用Github Copilot生成度量衡服务代码

除了国际化翻译之外，我们还需要做应用的本地化处理。例如：

提供一个公共的本地化组件，支持对数字、时间、度量衡在不同区域下的处理。

接下来分享团队基于Github Copilot开发副驾，示例完成以上代码的生成过程。



三、使用Github Copilot生成度量衡服务代码



三、使用Github Copilot生成度量衡服务代码

Prompt:

请用C#生成一个提供度量衡服务的实现类MeasurementService，它提供了以下方法将长度值转换为英寸、长度值转换为英尺、将长度值转换为英里、将长度值转换为厘米、将长度值转换为千米、将重量值转换为克、将重量值转换为千克、将功率值转换为瓦特、将电流值转换为安培、将电压值转换为伏特。例如将长度值转换为英寸的实现方法是public double ConvertToInch(double value, LengthUnit lengthUnit)，这个方法中遍历LengthUnit，做长度转换。方法请添加标准注释，使用中文注释。





zhouguoqing

Prompt: 请用C#生成一个提供度量衡服务的实现类MeasurementService, 它提供了以下方法将长度值转换为英寸、长度值转换为英尺、将长度值转换为英里、将长度值转换为厘米、将长度值转换为千米、将重量值转换为克、将重量值转换为千克、将功率值转换为瓦特、将电流值转换为安培、将电压值转换为伏特。例如将长度值转换为英寸的实现方法是public double ConvertToInch(double value, LengthUnit lengthUnit), 这个方法中将变量LengthUnit, 做长度转换。方法请添加标准注释, 使用中文注释。



GitHub Copilot



Used 1 reference >

Sure, here is a `MeasurementService` class in C# that provides the requested conversion methods. For the `ConvertToInch` method, I've added a simple switch statement to handle different `LengthUnit` values. You'll need to replace the comments with the actual conversion logic for the other methods. I've also added standard Chinese comments for each method.

```
public class MeasurementService
{
    /// <summary>
    /// 将长度值转换为英寸
    /// </summary>
    public double ConvertToInch(double value, LengthUnit lengthUnit)
    {
        switch (lengthUnit)
        {
            case LengthUnit.Meter:
                return value * 39.3701;
            case LengthUnit.Centimeter:
                return value * 0.393701;
            case LengthUnit.Kilometer:
                return value * 39370.1;
            case LengthUnit.Mile:
                return value * 63360;
            case LengthUnit.Foot:
                return value * 12;
        }
    }
}
```

Can you provide an example of how to use the `MeasurementService` class to convert a length value to inches?



三、使用Github Copilot生成度量衡服务代码

这里你会发现，其他方法未实现，需要继续告诉Github Copilot继续生成代码

Github Copilot生成的代码不一定全面准确，需要做代码确认，这个很关键

Prompt:

MeasurementService类中，
请参考ConvertToInch这个方法的实现方式和注释要求，
继续生成ConvertToMile, ConvertToFoot, ConvertToCentimeter,
ConvertToKilometer,
ConvertToGram, ConvertToKilogram,
ConvertToWatt, ConvertToAmpere,
ConvertToVolt等方法的具体转换逻辑，
每个方法都要实现代码输出



三、使用Github Copilot生成度量衡服务代码

首先选择整个类，然后输入以下Prompt

Prompt:

@workspace /tests 请对选中的代码，使用MSTest单元测试框架，生成单元测试代码，请为每个方法都实现单元测试





File Edit Selection View Go Run Terminal Help

CHAT: GITHUB COPILOT

GitHub Copilot

Hi @zhouguoqing, how can I help you?

zhouguoqing

@workspace /tests 请对选中的代码，使用MSTest单元测试框架，生成单元测试代码，请为每个方法都实现单元测试。

GitHub Copilot used @workspace

Used 1 reference >

Sure, here are the unit tests for your code:

```
[TestMethod]
public void ConvertToInch_ShouldConvertCorrectly_ForEachLengthUnit()
{
    Assert.AreEqual(1, _measurementService!.ConvertToInch(1, LengthUnit.Inch));
    Assert.AreEqual(39.3701, _measurementService!.ConvertToInch(1, LengthUnit.Meter));
    Assert.AreEqual(0.393701, _measurementService!.ConvertToInch(1, LengthUnit.Kilometer));
    Assert.AreEqual(39370.1, _measurementService!.ConvertToInch(1, LengthUnit.Millimeter));
    Assert.AreEqual(63360, _measurementService!.ConvertToInch(1, LengthUnit.Foot));
    Assert.AreEqual(12, _measurementService!.ConvertToInch(1, LengthUnit.Yard));
    Assert.AreEqual(36, _measurementService!.ConvertToInch(1, LengthUnit.Mile));
}

[TestMethod]
public void ConvertToMile_ShouldConvertCorrectly_ForEachLengthUnit()
{
    Assert.AreEqual(1, _measurementService!.ConvertToMile(1, LengthUnit.Mile));
    Assert.AreEqual(0.0000062137, _measurementService!.ConvertToMile(1, LengthUnit.Kilometer));
    Assert.AreEqual(0.0000157828, _measurementService!.ConvertToMile(1, LengthUnit.Meter));
    Assert.AreEqual(0.621371, _measurementService!.ConvertToMile(1, LengthUnit.Foot));
    Assert.AreEqual(1760, _measurementService!.ConvertToMile(1, LengthUnit.Yard));
}

T.Core.i10n.Service > MeasurementService.cs > ...
233 case CurrentUnit.Kiloampere:
234     return value * 1000;
235 default:
236     throw new NotSupportedException($"Unsupported current unit: {currentUnit}");
237 }
238 }
239
240 /// <summary>
241 /// 将电压值转换为伏特
242 /// </summary>
243 public double ConvertToVolt(double value, VoltageUnit voltageUnit)
244 {
245     // 根据电压单位进行转换
246     switch (voltageUnit)
247     {
248     case VoltageUnit.Volt:
249         return value;
250     case VoltageUnit.Millivolt:
251         return value / 1000;
252     case VoltageUnit.Microvolt:
253         return value / 1000000;
254     case VoltageUnit.Kilovolt:
255         return value * 1000;
256     default:
257         throw new NotSupportedException($"Unsupported voltage unit: {voltageUnit}");
258     }
259 }
260 }
261 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS .NET Test Log

Ln 261, Col 2 (9004 selected) Spaces: 4 UTF-8 CRLF C# using system



今天我们共同探讨了基于AIGC实现.NET应用国际化

从智能翻译到代码生成，这是LLM时代一个小小的案例，但是

未来：

有LLM加持的智能翻译将更精准，全面提升用户体验。

代码自动生成将全面释放开发者创造力。

随着AIGC的迭代升级，AI将为我们带来更多应用创新和价值创造。

感谢大家的聆听。

