คู่มือสำหรับนักพัฒนา: โปรเจก Akkhie Multipurpose

1. บทนำ (Introduction)

เอกสารนี้เป็นคู่มือสำหรับนักพัฒนาเพื่อทำความเข้าใจสถาปัตยกรรม โครงสร้าง และส่วนประกอบหลักของโปรเจก **Akkhie Multipurpose** ซึ่งเป็นแอปพลิเคชัน Desktop สำหรับ Windows (Windows Forms) ที่พัฒนาด้วยภาษา C# บน .NET Framework

วัตถุประสงค์ของโปรแกรม:
รวบรวมเครื่องมืออำนวยความสะดวกและเครื่องมือแก้ปัญหา (Troubleshooter) ต่างๆ ไว้ในที่เดียว เพื่อให้ง่ายต่อการใช้งานและบำรุงรักษา
เทคโนโลยีหลักที่ใช้:

- **ภาษา:** C#
- **UI Framework:** Windows Forms (.NET Framework)
- **ฐานข้อมูล:** มีการเชื่อมต่อกับ SQL Server (จาก Microsoft.Data.SqlClient)
- **อื่น ๆ:**
  - Newtonsoft.Json: สำหรับการอ่านไฟล์ shortcuts.json
  - CsvHelper: สำหรับการอ่านไฟล์ .csv ในโฟลเดอร์ Licenses

2. ภาพรวมสถาปัตยกรรม (Architecture Overview)

แอปพลิเคชันใช้สถาปัตยกรรมที่เรียบง่ายแต่ยืดหยุ่น โดยมีองค์ประกอบหลักดังนี้:

- **Main Form (Form1.cs):** เป็นหน้าต่างหลักของโปรแกรม ทำหน้าที่เป็น Container สำหรับแสดง User Control ต่างๆ ที่เป็นฟังก์ชันหลักของโปรแกรม
- **User Controls:** แต่ละฟังก์ชันหลักของโปรแกรมจะถูกสร้างเป็น User Control แยกกัน เพื่อให้ง่ายต่อการจัดการและนำไปใช้ใหม่ User Control หลักๆ ได้แก่:
  - TroubleshooterControl: หน้าสำหรับรวมเครื่องมือแก้ปัญหาต่างๆ
  - OfficeToolsControl: เครื่องมือเกี่ยวกับ Microsoft Office
  - WindowsSettingsControl: เครื่องมือตั้งค่า Windows
  - ShippingCostControl: เครื่องมือคำนวณค่าขนส่ง
  - และอื่นๆ
- **Troubleshooter Tools:** เป็นหัวใจสำคัญของโปรแกรมในส่วนแก้ปัญหา โดยใช้ Interface ชื่อ ITroubleshooterTool เป็นต้นแบบ ทำให้สามารถเพิ่มเครื่องมือใหม่ๆ เข้าไปในระบบได้อย่างง่ายดายโดยไม่ต้องแก้ไขโค้ดหลัก

3. การตั้งค่าโปรเจก (Project Setup)

1. เปิดไฟล์ Multipurpose.sln ด้วย Visual Studio (แนะนำเวอร์ชัน 2019 หรือใหม่กว่า)
2. Restore NuGet Packages: คลิกขวาที่ Solution ใน Solution Explorer แล้วเลือก "Restore NuGet Packages" เพื่อให้ Visual Studio ดาวน์โหลด Dependencies ที่จำเป็น (เช่น Newtonsoft.Json, CsvHelper, Microsoft.Data.SqlClient)

3. ตั้งค่า Connection String: ในไฟล์ App.config จะมี connectionStrings ซึ่งจำเป็นต้องตั้งค่าให้ถูกต้องเพื่อเชื่อมต่อกับฐานข้อมูลที่เครื่องมือ Troubleshooter ต่างๆ ต้องใช้งาน
   ```
   <connectionStrings>
     <add name="DefaultConnection" connectionString="Data Source=YOUR_SERVER;Initial Catalog=YOUR_DATABASE;Integrated Security=True;" providerName="Microsoft.Data.SqlClient" />
   </connectionStrings>
   ```
4. กด F5 เพื่อ Build และ Run โปรแกรม

4. ส่วนประกอบหลัก (Core Components)

4.1. Main Form (Form1.cs)

เป็นหน้าต่างหลักที่ใช้ TabControl ในการสลับการแสดงผลของ User Control ต่างๆ ทำให้ผู้ใช้สามารถเลือกใช้เครื่องมือที่ต้องการได้สะดวก

4.2. User Controls

แต่ละ User Control จะรับผิดชอบฟังก์ชันการทำงานเฉพาะด้าน:

- **TroubleshooterControl.cs**:
  - **หน้าที่:** แสดงรายการเครื่องมือแก้ปัญหาทั้งหมดที่มีในระบบ
  - **การทำงาน:** เมื่อ Control นี้โหลดขึ้นมา จะใช้ Reflection ในการค้นหา Class ทั้งหมดในโปรเจกที่ implement ITroubleshooterTool แล้วนำชื่อและคำอธิบายมาแสดงใน ComboBox ให้ผู้ใช้เลือก
  - เมื่อผู้ใช้เลือกเครื่องมือและกดปุ่ม "Execute" ระบบจะสร้าง Instance ของเครื่องมือที่เลือกและเรียกใช้เมธอด Execute()
- **OfficeToolsControl.cs**:
  - **หน้าที่:** จัดการเกี่ยวกับการติดตั้งและเปิดใช้งาน Microsoft Office
  - **การทำงาน:** อ่านข้อมูล License Key จากไฟล์ Licenses/LicenseOffice.csv มาแสดงให้ผู้ใช้เลือก และมีฟังก์ชันสำหรับรันคำสั่ง Command-line เพื่อติดตั้ง Key หรือตรวจสอบสถานะ
- **WindowsUpgradeControl.cs / WindowsSettingsControl.cs**:
  - **หน้าที่:** จัดการเกี่ยวกับการอัปเกรดและตั้งค่า Windows
  - **การทำงาน:** คล้ายกับ OfficeToolsControl โดยจะอ่านข้อมูล License Key จาก Licenses/LicenseWindows.csv และรันคำสั่งที่เกี่ยวข้อง
- **ShippingCostControl.cs**:
  - **หน้าที่:** เครื่องมือคำนวณค่าขนส่งสินค้า
  - **การทำงาน:** มี UI สำหรับกรอกข้อมูลที่จำเป็นในการคำนวณ และแสดงผลลัพธ์

4.3. Troubleshooter Tools (ในโฟลเดอร์ Tools)

เป็นส่วนที่สำคัญและยืดหยุ่นที่สุดของโปรแกรม

- **Interface: ITroubleshooterTool.cs**
  - เป็นสัญญา (Contract) ที่เครื่องมือแก้ปัญหาทุกตัวต้อง implement ประกอบด้วย:
    - ToolName (string): ชื่อของเครื่องมือที่จะแสดงให้ผู้ใช้เห็น
    - ToolDescription (string): คำอธิบายการทำงานของเครื่องมือ
    - Execute(string connectionString, object[] parameters): เมธอดหลักในการทำงานของเครื่องมือ โดยรับ Connection String และพารามิเตอร์อื่นๆ ที่จำเป็น
- **ตัวอย่าง Tools ที่มีอยู่:**
  - UnlockQuotationTool.cs: ปลดล็อกใบเสนอราคาในฐานข้อมูล
  - ChangeQuotationTool.cs: แก้ไขข้อมูลใบเสนอราคา
  - FixShippingCostTool.cs: แก้ไขค่าขนส่งที่ไม่ถูกต้องในฐานข้อมูล
  - UpdateAddressTool.cs: อัปเดตที่อยู่ลูกค้า
  - NewWasteTool.cs: เพิ่มข้อมูลของเสียใหม่เข้าระบบ
- **Class: ToolHelpers.cs**
  - เป็น Static Class ที่รวบรวมเมธอดช่วยเหลือต่างๆ ที่ใช้บ่อยใน Tools เช่น การ Execute คำสั่ง SQL

5. การเพิ่มฟังก์ชันใหม่ (Adding New Features)

การเพิ่มเครื่องมือ Troubleshooter ใหม่

นี่คือจุดเด่นของสถาปัตยกรรมนี้ คุณก้องสามารถเพิ่มเครื่องมือใหม่ได้ง่ายๆ โดยทำตามขั้นตอนต่อไปนี้:

1. **สร้าง Class ใหม่:** ในโฟลเดอร์ Tools สร้าง C# Class ใหม่ (เช่น MyNewAwesomeTool.cs)
2. **Implement Interface:** ให้ Class ใหม่ implement ITroubleshooterTool

```
using System;
// ... other using statements

namespace Multipurpose.Tools
{
    public class MyNewAwesomeTool : ITroubleshooterTool
    {
        public string ToolName => "My New Awesome Tool"; // ชื่อเครื่องมือ

        public string ToolDescription => "This tool does something awesome."; // คำอธิบาย

        // เมธอดหลักในการทำงาน
        public bool Execute(string connectionString, object[] parameters)
        {
```

```
            try
            {
                // เขียนโค้ดการทำงานของเครื่องมือที่นี่
                // เช่น เชื่อมต่อฐานข้อมูล, แก้ไขข้อมูล, ฯลฯ
                // ใช้ connectionString ที่รับเข้ามา

                // ตัวอย่างการเรียกใช้ Helper
                // ToolHelpers.ExecuteNonQuery(connectionString, "UPDATE MyTable
SET ...", null);

                return true; // คืนค่า true ถ้าทำงานสำเร็จ
            }
            catch (Exception ex)
            {
                // จัดการ Error และคืนค่า false
                Console.WriteLine(ex.Message);
                return false;
            }
        }
    }
}
```

3. **เสร็จสิ้น!** เพียงเท่านี้ เมื่อคุณรันโปรแกรม TroubleshooterControl จะค้นหาและแสดงเครื่องมือใหม่ของคุณใน ComboBox โดยอัตโนมัติ ไม่จำเป็นต้องแก้ไขโค้ดในส่วน UI เลย

การเพิ่ม User Control ใหม่

1. สร้าง User Control ใหม่ในโปรเจก
2. ออกแบบ UI และเขียนโค้ดสำหรับฟังก์ชันที่ต้องการ
3. ไปที่ Form1.cs [Design] แล้วเพิ่ม TabPage ใหม่ใน TabControl
4. ลาก User Control ที่สร้างใหม่มาวางใน TabPage นั้น

6. โค้ดสำคัญ (Key Code Snippets)

TroubleshooterControl.cs - การค้นหา Tools ด้วย Reflection

```
private void LoadTools()
{
    var toolType = typeof(ITroubleshooterTool);
    var tools = AppDomain.CurrentDomain.GetAssemblies()
        .SelectMany(s => s.GetTypes())
        .Where(p => toolType.IsAssignableFrom(p) && !p.IsInterface);
```

```
    foreach (var tool in tools)
    {
        // สร้าง instance ชั่วคราวเพื่อเอาชื่อและคำอธิบาย
        var instance = (ITroubleshooterTool)Activator.CreateInstance(tool);
        comboBoxTools.Items.Add(instance);
    }
    // ... ตั้งค่า DisplayMember และ ValueMember ของ ComboBox
}
```

โค้ดส่วนนี้จะทำการค้นหา (Scan) ทุก Class ที่สืบทอดมาจาก ITroubleshooterTool แล้วนำมาใส่ใน ComboBox โดยอัตโนมัติ