DN8

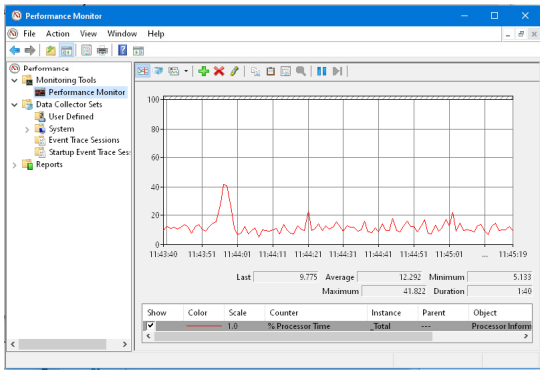
Zürcher Hochschule für Angewandte Wissenschaften

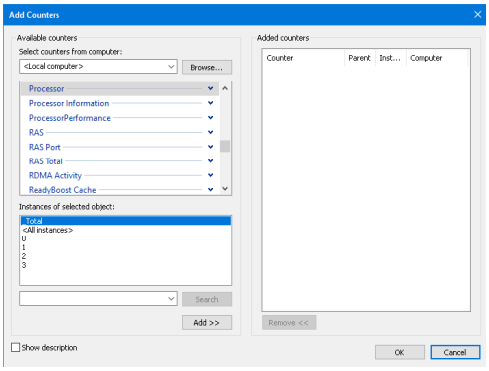
Arbeitsblatt: DNET1

Name: _____ **Kurznamen:** _____

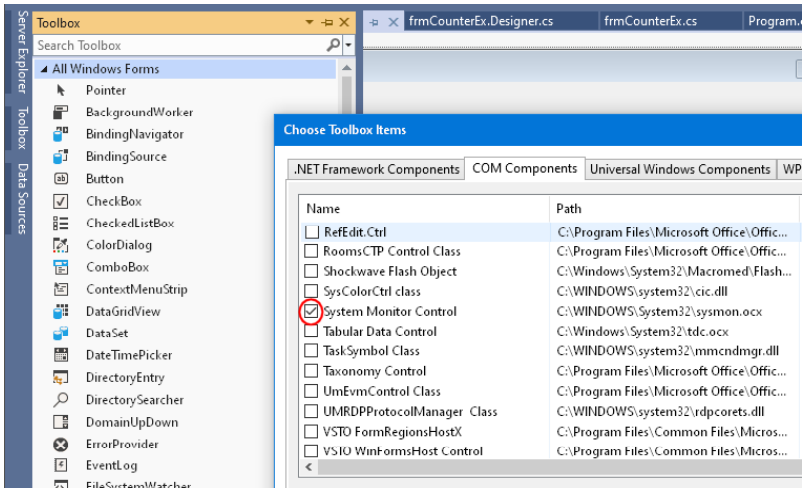
Performance Monitor COM Control

Der Performance Monitor ist ein System Werkzeug um den Auslastungszustand der eigenen oder einer entfernten Maschine anzuzeigen. Dieser kann z.B. über die Kommandozeile mittels `perfmon` gestartet werden. Danach können durch Drücken des grünen "+" Knopfes beliebige Performance Counter hinzugefügt werden.





Der Performance Monitor GUI ist selber ein sog. COM Control das mittels Rechtsklick in die Toolbox → Choose Items… → COM Components → System Monitor Controls dem Visual Studio hinzugefügt werden kann (wird unten bei der Toolbox angehängt).



Das Monitor Control kann danach im GUI Builder in .NET Forms verwendet werden. Die Darstellungseigenschaften können auch programmatisch vorgenommen werden . z:B.

```
private void InitializeSystemMonitor() {
            axSystemMonitor1.BackColor = System.Drawing.Color.LightGray;
```

```
            axSystemMonitor1.GridColor = System.Drawing.Color.Gray;
            axSystemMonitor1.ShowToolbar = false;
            axSystemMonitor1.ShowValueBar = false;
            axSystemMonitor1.ShowTimeAxisLabels = false;
            axSystemMonitor1.ShowVerticalGrid = true;
            axSystemMonitor1.ShowHorizontalGrid = true;
            axSystemMonitor1.ShowLegend = false;
            axSystemMonitor1.ChartScroll = true;
}
```

Aufgabe 1

Erstellen Sie ein Windows Forms (Framework) GUI, das in einem System Monitor Control die aktuelle totale CPU Auslastung darstellt

Hinweise:

- Ein .NET Framework Projekt erstellen
- Der Prozessor Time Counter dem Monitor Control hinzufügen:

```
ICounterItem item;
axSystemMonitor1.AddCounter("\\Processor(_Total)\\% Processor Time", out item);
```

Aufgabe 2

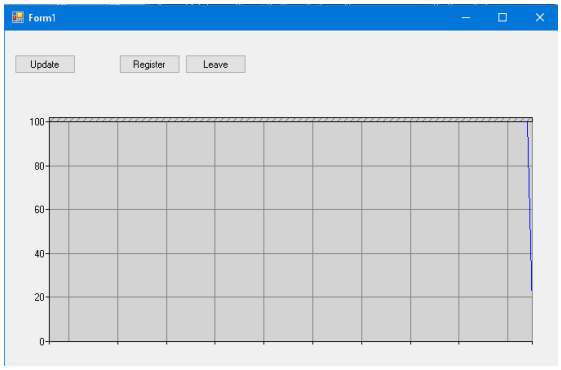
Auch zusätzliche Werte können über eigenen Performance Counter angezeigt werden. Anbei ein Beispiel eines Counters der Kategorie "Button Usage" mit dem Performancecounter "UpdateClick"

```
using System.Diagnostics;
using SystemMonitor;

public class ButtonUsageCounter : IDisposable {
        public const string CategoryName = "Button Usage";
        public PerformanceCounter UpdateClicks;

        public ButtonUsageCounter() {
            Dispose();
            if (!PerformanceCounterCategory.Exists(CategoryName)) {

                CounterCreationData UpdateButtonClick = new CounterCreationData();
                UpdateButtonClick.CounterName = "UpdateClick";
                UpdateButtonClick.CounterType = PerformanceCounterType.NumberOfItems32;

                CounterCreationDataCollection ClickCounters
                      = new CounterCreationDataCollection();
                ClickCounters.Add(UpdateButtonClick);

                //Create new Performance counter Category
                PerformanceCounterCategory.Create(CategoryName, "",
                    PerformanceCounterCategoryType.SingleInstance, ClickCounters);

                // Now create actual measuring counters
                UpdateClicks = new PerformanceCounter(CategoryName, "UpdateClick", false);
            }
        }
       public void Dispose() {
            if (PerformanceCounterCategory.Exists(CategoryName)) {
                PerformanceCounterCategory.Delete(CategoryName);
            }
       }
}
```

Aufgabe

Fügen Sie einen "Update" Knopf dem GUI hinzu, der den Performance Counter jeweils um eins erhöht. Fügen Sie weiter der Klasse ButtonUsageCounter einen zweiten Performance Counter "RegisterClick" hinzu, der über den "Register" Knopf um eins

erhöht und dem "Leave" Knopf um eins vermindert wird. Stellen Sie die Werte dieser beiden Performance Counter im Monitor GUI dar.



Hinweise:

- Der Counter Wert kann wiefolgt erhöht werden

```
ButtonUsageCounter  counter = new ButtonUsageCounter();
counter.UpdateClicks.Increment();
```

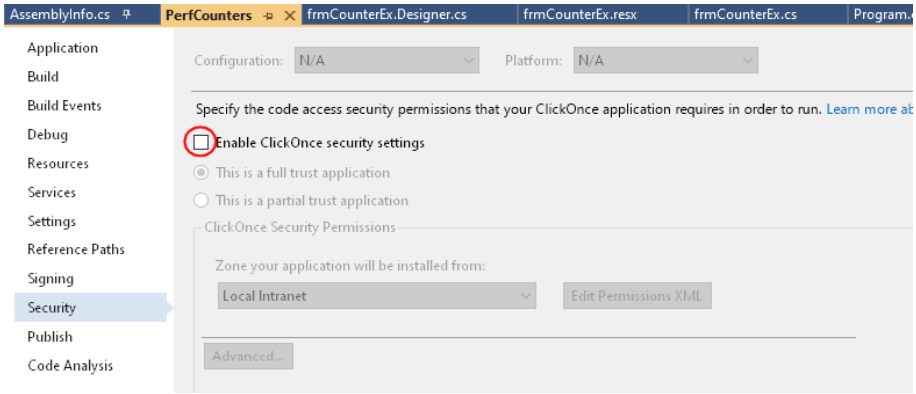
- Den Button Usage Counter dem Monitor Control hinzufügen:

```
ICounterItem item;
axSystemMonitor1.AddCounter("\\Button Usage\\*" ", out item);
```

- Es muss im app.manifest folgende Permission im Security Teil gesetzt werden

```
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <!-- UAC Manifest Options -->
      <requestedExecutionLevel level="requireAdministrator" uiAccess="false" />
      </requestedPrivileges>
      <applicationRequestMinimum>
        <defaultAssemblyRequest permissionSetReference="Custom" />
<PermissionSet class="System.Security.PermissionSet" version="1" ID="Custom"
SameSite="site" Unrestricted="true" />
      </applicationRequestMinimum>
    </security>
</trustInfo>
```

Weiter muss in den Projekteinstellung die ClickOnce Security Setting deaktiviert werden.



Abgabe:

Praktikum: DN8.1

Datei: ButtonUsageCounter.cs

Aufgabe 2 – Low Level Programmierung

Für die Kalenderfunktion ist die richtige Uhrzeit und Datum eine notwendige Vorraussetzung. Leider sind die Uhrenbausteine der meisten Computer ziemlich ungenau und sie sollte deshalb periodisch überprüft und allenfalls korrigiert werden - was moderne Betriebssystem ja in der Regel automatisch machen. Es gibt im Internet für diesen Zweck sogenannte NIST Internet Time Server, die auf Anfrage die exakte Zeit liefern. Eine Zusammenstellung der verfügbaren Time Server findet man unter http://tf.nist.gov/tf-cgi/servers.cgi#

Leider kann in .NET nicht direkt die Uhr der eigenen Maschine gesetzt werden. Es gibt jedoch in der Kernel32.dll einen entsprechenden Aufruf; siehe Zusatzdokument (pdf)

Schreiben Sie eine Methode, welche die Uhr mit der NIST Zeit synchronisiert.

Hinweis: es muss folgendes `struct` mit den zu C++ analogen Datentypen als **ref** dem Aufruf übergeben werden.

```
public struct SystemTime                        // win32 struct
{
      public short wYear;
      public short wMonth;
      public short wDayOfWeek;
      public short wDay;
      public short wHour;
      public short wMinute;
      public short wSecond;
      public short wMilliseconds;
};
```

Hinweise

- Sie müssen den Aufruf unter Admin Rechten ausführen;
  - Rechts-Klick und runas
  - Während Entwicklung: VS als Admin starten
- Falls Probleme mit **`SetSystemTime`** auftreten, dann versuchen Sie **`SetLocalTime`** (gleiche Schnittstelle)

Abgabe:

Praktikum: DN8.2

Datei: NISTTime.cs

Weitere Informationen

Custom Perfomance Counter
https://coding-examples.com/csharp/creating-custom-performance-counter-in-c/

Logman und Co
https://www.dell.com/support/kbdoc/de-ch/000195727/capture-performance-logs-using-logman-exe-relog-exe-and-typeperf-exe

Beispiel der Einbindung des Controls
https://www.codeproject.com/Articles/38347/Take-advantage-of-the-Windows-Vista-built-in-Syste