ITKPU Lab 4: Use of Assemblies

Formål:

At opnå erfaring med udvikling og brug af delte assemblies samt ikke-visuelle komponenter i .Net.

Forudsætninger

At du har læst om Componenter i .Net samt PowerPoint præsentationerne til lektionerne Life-cyclemanagedment samt versionering og probing.

I denne opgave skal der udvikles en komponent (non-UI) som indeholder en algoritme til kontrol af et CPR-nummers gyldighed. Komponenten udvikles i 3 forskellige udgave som anført i delopgaverne.

Delopgave 1

I denne delopgave skal der laves en privat assembly (dll) som indeholder klassen CprCheck med funktionen Check.

Fremgangsmåde:

1. Lav en ny tom "solution" og tilføj et nyt C# projekt af typen "Class Library". Giv det f.eks. navnet CprDLL.
2. Omdøb klassen fra Class1 til CprCheck.
3. Tilføj en funktion med følgende signatur:
   public bool Check(string cprTxt, out CprError error)
   hvor CprError er en opremsningstype (enum):
   public enum CprError {NoError, FormatError, DateError, Check11Error};
4. Implementer funktionen Check
   Denne bør opbygges a 3 private funktion som udfører hver deres check. Gyldigt format (10 cifre), gyldig dato (brug DateTime klassen til dette) og 11test.

```
/// <summary>
/// The CPR check sum algorithm is calculated by mulitiplying each digit with a factor
/// and then add all results and divide the sum by 11.
/// Factors: 4327654321
/// CPR: 0609240121
/// Sum: 0 + 18 + 0 + 72 + 12 + 20 + 0 + 1 + 4 + 1 = 121 / 11 = 11.0 -> CPR is OK
/// </summary>
///
private CprError Check11Test(string cprTxt)
{
  CprError cprError = CprError.NoError;

  int sum = 0;
  for (int i = 0; i < 3; i++)
    sum += int.Parse(cprTxt.Substring(i, 1)) * (4 - i);
  for (int i = 3; i < 10; i++)
    sum += int.Parse(cprTxt.Substring(i, 1)) * (10 - i);
  if (sum % 11 != 0)
    cprError = CprError.Check11Error;
}
```

Delpgave 2

Lav et Windows WPF testprogram hvor man kan indtaste et cpr-nummer i en TextBox, og så ved tryk på en kontrol-knap kan få kontrolleret om det indtastede CPR-nummer er gyldigt. Programmet skal benytte DLL'en fra delopgave 1 som en "private assembly" til at udfører kontrollen.

Delopgave 3

I denne delopgave skal der laves en shared assembly (dll) som indeholder klassen CprCheck med funktionen Check ganske som delopgave 1. Men denne gang skal assembly'en have et "strong name" så den kan installeres i GAC'en (Global Assembly Cache).

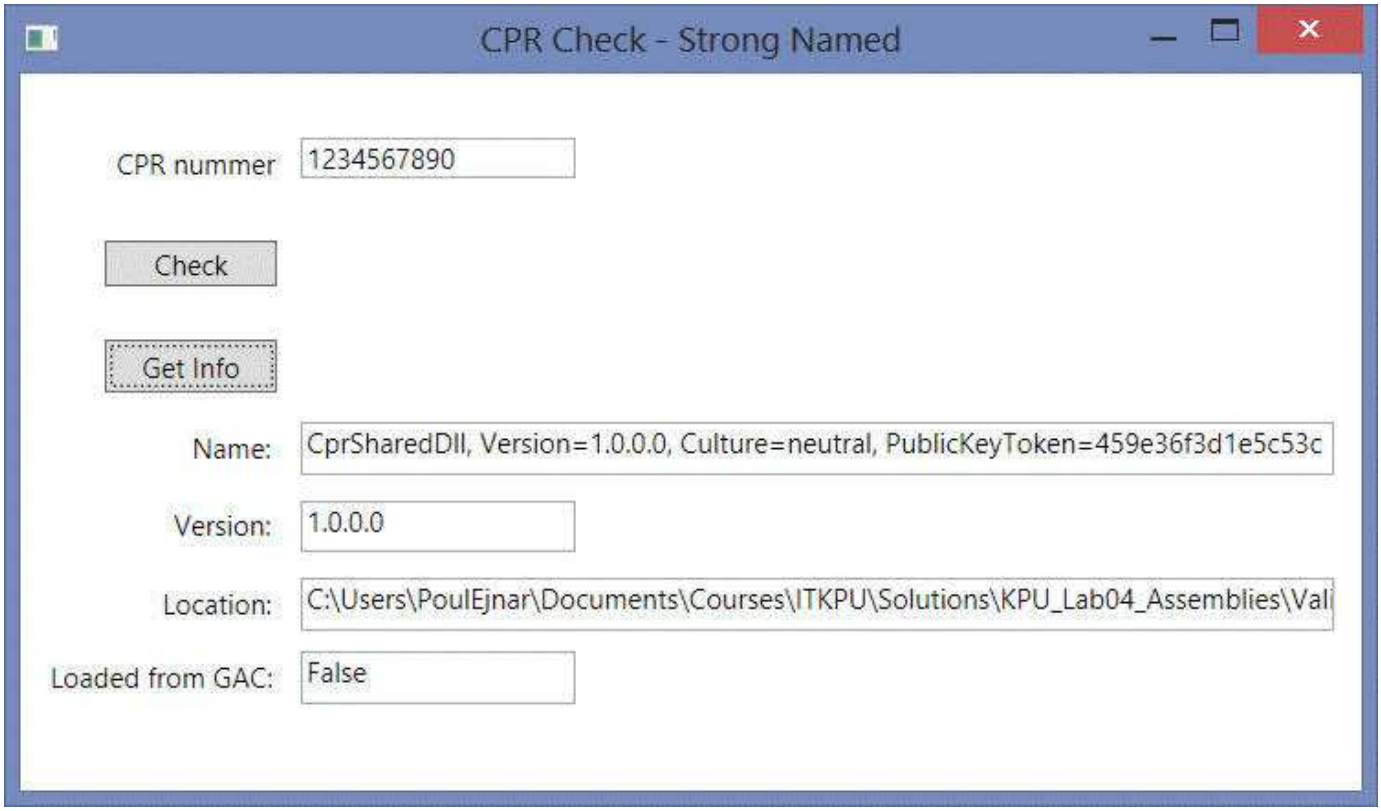
Fremgangsmåde:

1. Tilføj et nyt C# projekt af typen "Class Library". Giv det f.eks. navnet CprSharedDLL.
2. Kopier implementering fra delopgave 1 hertil.
3. Giv assemblyen et strong name ved at signere assemblyen.

Delopgave 4

Lav et nyt Windows WPF testprogram som er en kopi af delopgave 2, men som bruger en assembly med et "strong name" - den fra delopgave 3. Prøv at ligger assemblyen lokalt, og prøv at installerer assemblyen i GAC'en, og test at programmet loader den derfra. På Windows 7 og 8 skal installation i GAC'en ske med konsolprogrammet GacUtil (GacUtil –i <assemblyName>. På XP og Vista dette kan gøres ved at trækken DLL'en overpå mappen Assembly som ligger i systemmappen (C:\Windows).

Til at teste hvorfra assembly'en er loaded kan man bruge reflection (Type assType = cprNo.GetType(); // assType indeholder nu alt den info vi ønsker).



Delopgave 5 - WPF Toolbox Control

Ved at lave CPR-kontrollen som en WPF Toolbox Control så kan man tilføje kontrollen til toolboxen i Visual Studio. Dette er meget smart, hvis man har lavet en god kontrol, som kan bruges af andre.

For at kunne udvikle en WPF Toolbox Contro så er det nødvendigt at installerer en bestemt SDK til Visual Studio: Visual Studio 2013 SDK http://www.microsoft.com/en-us/download/details.aspx?id=40758

Du kan få et overblik ved at læse ”How to: Create a Toolbox Control That Uses WPF”: http://msdn.microsoft.com/en-us/library/vstudio/ee712574.aspx

Eller endnu bedre: “Walkthrough: Creating a WPF Toolbox Control”: http://msdn.microsoft.com/en-us/library/vstudio/ee712573.aspx