Grobkonzept KLib

# Grundsätzliches

# Module

* 01Base: Basisfunktionalitäten ohne GUI  
  NS: KLib.Base, Abhängigkeiten: Keine
* 02Sql: Datenbankabstraktion  
  NS: KLib.Sql, Abhängigkeiten: 01Base
* 05ToolsForms: Anbindung an Windows Forms  
  NS: KLib.Forms, Abhängigkeiten: 01Base
* 05WpfTools: Anbindung an WPF  
  NS: KLib.Wpf, Abhängigkeiten: 01Base
* 06WpfSqlTools: Wpf Tools bzgl. Datenbankanbindung  
  NS: KLib.WpfSql, Abhängigkeiten: 02Sql, 05WpfTools

# Templates

Im Verzeichnis Templates liegen Vorlagen für Konfigurationen, etc.

* Template\_log4net.config: Konfiguration (und Dokumentation!) zur Verwendung des log4net Logging Frameworks.
* Template\_Database.config: Konfigurationsbeispiel für Datenbankanbindung.

# HowTo’s

## Neue App mit KLib starten

1. Neue WPF Anwendung erzeugen
2. Neues Projektverzeichnis dazugeben und dort die benötigten Module bzw. Projekte aus KLib dazugeben
3. Abhängigkeit der App zum „höchsten“ KLib Modul definieren.
4. Unter „Verweise“ die Module der KLib aus der Projektmappe dazuhängen
5. Build…

## Log4net einbinden

1. Im NuGet Manager der Projektmappe log4net installieren (für die App)
2. Die KLib verwendet auch log4net: Konflikte!?!?!?!
3. Im Template Verzeichnis gibt es eine Vorlage (template.log4net.config).
4. Irgendwo außerhalb einer Klasse (z.B. in program.cs) log4net konfigurieren über Attribute:  
   [assembly: log4net.Config.XmlConfigurator(Watch = true, ConfigFile="log4net.config")]  
   Damit wird die eigentliche Konfig aus der Datei log4net.config gelesen (XML Konfig). Fehlt die Angabe von ConfigFile=…, so wird die Konfig aus der app.xml gelesen.  
   ODER (besser):  
   [assembly: log4net.Config.XmlConfigurator(Watch = true, ConfigFileExtension="log4net.config")]  
   Hier wird die Konfig in der Datei PROGRAMMNAME.exe.log4net.config gesucht.
5. Im C# Code einen Logger konfigurieren:  
     
   static readonly log4net.ILog Logger = log4net.LogManager.GetLogger(System.Reflection.MethodBase.GetCurrentMethod().DeclaringType);
6. Das Logging Window aus der KLib verwenden:  
     
   if (!WndLogger.IsAlreadyOpen())  
   {  
    WndLogger w = WndLogger.CreateOrGetWndLogger(this);  
    w.Top = Top;  
    w.Left = Left + ActualWidth;  
    w.Height = ActualHeight;  
    w.Show();  
   }

## Wpf Style mit MahApps.Metro aufpeppen

1. Im NuGet Manager der Projektmappe nach mahapps suchen und MahApps.Metro installieren (nur für die neue App nicht für KLib Module)
2. Im xaml-File (Window):
   1. Namespace dazugeben:  
      xmlns:Metro="clr-namespace:MahApps.Metro.Controls;assembly=MahApps.Metro"
   2. Den <Window Tag ändern in <Metro:MetroWindow
   3. Code Behind:  
      using MahApps.Metro.Controls;  
      …  
      public partial class MainWindow : MetroWindow
   4. Window Frame aktivieren:  
      BorderBrush="{DynamicResource AccentColorBrush}"  
      BorderThickness="1"
3. App.xaml: Ressource Dictionary ergänzen (unter <Application.Ressources>:  
   <ResourceDictionary>  
    <ResourceDictionary.MergedDictionaries>  
    <!-- MahApps.Metro resource dictionaries. Make sure that all file names are Case Sensitive! -->  
    <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Controls.xaml" />  
    <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Fonts.xaml" />  
    <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Colors.xaml" />  
    <!-- Accent and AppTheme setting -->  
    <!-- <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Blue.xaml" /> -->  
    <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Cobalt.xaml" />  
    <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseLight.xaml" />  
    </ResourceDictionary.MergedDictionaries>  
   </ResourceDictionary>

## Windows Service erzeugen

1. Neues Projekt anlegen vom Typ “Windows-Dienst”
2. Im generierten Code (Service Dispatcher für Service1) den folgenden Block als Installer dazugeben:

[RunInstaller(true)]

pubic class MyInstaller : Installer

{

private ServiceInstaller m\_ThisService;

private ServiceProcessInstaller m\_ThisServiceProcess;

public MyInstaller()

{

m\_ThisService = new ServiceInstaller();

m\_ThisServiceProcess = new ServiceProcessInstaller();

m\_ThisServiceProcess.Account = ServiceAccount.LocalSystem;

m\_ThisService.ServiceName = "BTA ASD Services";

m\_ThisService.Description = "Services of the ASD Data Collector";

m\_ThisService.StartType = ServiceStartMode.Automatic;

Installers.Add(m\_ThisService);

Installers.Add(m\_ThisServiceProcess);

}

}

1. Unter Template/ServiceControl befindet sich das InstallUtil.exe von Microsoft zum Installieren des Services (siehe .bat Datei für Details)

## Datenbankunterstützung (KLib.Sql)

1. .net Datenbanktreiber per NuGet installieren (z.B. postgres: npgsql)
2. Die Anwendung mittels Assembly Attribute konfigurieren (analog log4net):  
   [assembly: KLib.Sql.XmlConfigurator(ConfigFile = "../conf/Database.config")]
3. Konfigurationsdatei aus Template/ kopieren und entsprechend anpassen.
4. Beispiel Code:

DbConnection conx = DbManager.getConnectionManager().getConnection();

if (conx.open())

{

object r = conx.execSQL\_select("select count(\*) from machines");

conx.close();

}

## OxyPlot Wpf Chart einhängen

1. Über NuGet OxyPlot.Wpf installieren
2. ViewModel Klasse anlegen:  
    class MainWindowModel  
    {  
    public PlotModel MyModel { get; private set; }  
     
    public MainWindowModel()  
    {  
    this.MyModel = new PlotModel { Title = "Example 1" };  
    this.MyModel.Series.Add(new FunctionSeries(Math.Cos, 0, 10, 0.1, "cos(x)"));  
    }  
    }
3. Window.xaml  
   Namespace dazugeben: xmlns:oxy=<http://oxyplot.org/wpf>  
   DataContext setzen (local ist der Namespace der eigenen Anwendung):  
    <Window.DataContext>  
    <local:MainWindowModel/>  
    </Window.DataContext>  
   Content dazu:  
   …  
   <oxy:PlotView Model="{Binding MyModel}"/>  
   …