TODO

done általánosságban MVVM

miért mvvm

osztály leírások

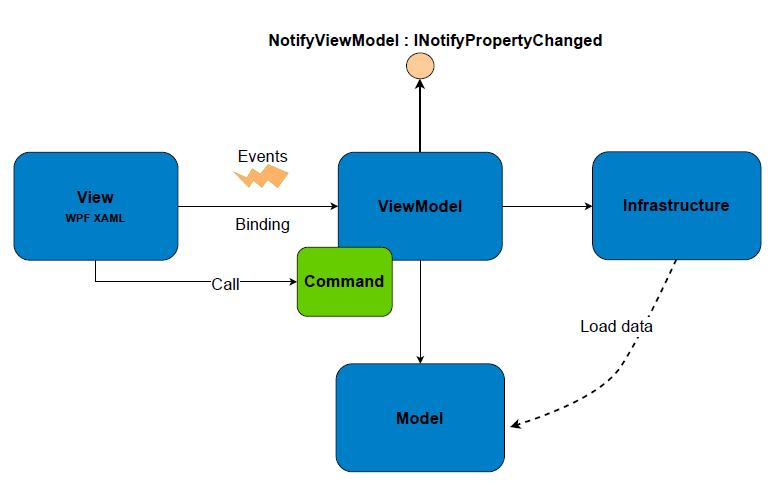
----

Mvvm: Model-View-ViewModel (MVVM) minta

Az MVVM a szoftferfejlesztésben használt minta, amely Microsoft által let kifejlesztve specifikusan, Presentation mintát alapúl véve specializálódott (állapotot és viselkedést reprezentálja függetlenül a felhasználói grafikus felülettől).

Az MVVM alapja a Model-View-Controller (MVC), olyan modern UI platformokra írányul, mint (WPF és Silverlight), amelyekben van UX fejlesztési lehetőség azaz más követelményei vannak, mint egy "hagyományosabb" felület fejlesztésének.[1]

Az MVVM az ügyfélalkalmazások létrehozásának egyik módja, amely kihasználja a WPF platform alapvető jellemzőit, lehetővé teszi az alkalmazás funkcióinak egyszerű tesztelését, és segíti a fejlesztőket és a tervezőket a így kevesebb a műszaki nehézség az együttmőködésben.



View: A nézetet az XAML határozza meg, és a kód mögött nem lehet logika. A nézetmodellhez csak az adatok kötésével kötődik, úgynevezett Data Binding (egy állandó kapcsolatot hoz ).

Model: A modell felelős az adatok olyan módon történő tárolásáért, amelyet a WPF könnyen felhasználhat. Megfelelően az INotifyPropertyChanged és / vagy INotifyCollectionChanged eseményeket kell végrehajtania.

ViewModel: A ViewModel egy specifikus modell kimondottan a nézethez az alkalmazásban, vagy mondhatjuk a nézet absztrakciójaként is. A nézet szempontjából releváns adatokat használja és a nézetek viselkedését, általában parancsokkal köti össze.

**[1] Mastering Windows Presentation Foundation**. Sheridan Yuen. February 17, 2017

Helper

public static class GeneralMessageCollection

public static string GeneralMessageRecived(string customText)

public static string GeneralMessageRecived(string customText, XmlFilter xmlData)

public static string LogIntoFile(Exception ex)

public static string IsRunningStateChecker(bool state)

Infrastructure

public class MessageBoxWrapper

public static MessageBoxResult Show(string msg, string title, MessageBoxButton buttonStyle, MessageBoxImage image)

public class RelayCommand : ICommand

Message

//A bejövő üzenetek üzeteket rendezi egyetlen tömbe, amit az összes osztályból elérhető.

public static class ByteMessageBuilder

public static void SetByteArray(int index, byte inputByte)

public static void SetByteArray(int index, string inputByte)

public static void SetByteIncomingArray(int index, string inputByte)

public static void ResetByteIncomingArray()

public static string[] GetByteIncomingArray()

public static byte[] GetByteArray()

public static byte ConvertStringToByte(string strByte)

public static string ConvertDecimalStringToHexString(string decimalNumber)

//A bejövő üzenetek validásáért felelős osztly.

public static class ValidatorIncomingMessage

private static bool CheckRightEOF(string incomingByte, XmlFilter xmlFilter)

public static bool ErrorMessageBack(XmlFilter xmlData, string incomingData)

public static bool ValidationEOF(XmlFilter xmlData)

Report

//A mérések elredményeinek elmentése excel fájlba.

public class ReportCreator

public void AddData(List<string> data, List<List<string>> data2, string name, List<string> measuredCardsColum)

private void FixExtraValues()

public void CreateFile()

private void ColorChanger(int row, int colum, ExcelWorksheet excelWorksheet, Color cellBackGroundColor)

private void LoopThrough(ExcelWorksheet workSheet)

private void OtherCellsModification(ExcelWorksheet workSheet)

XmlHandler

//A XML fájlok tartarmára vonatkozó szürőket tartalmazza, illetve a bejövő üzeneteket fordítja le byte-ról .

public sealed class XmlFilter

public static XmlFilter Instance

public List<string> GetCardTypeNames()

public List<string> GetMeasurements(string cardType)

public List<string> GetMeasurementsWithoutAutoMeasure(string cardType)

public string GetSelectedCardTypeValue(string cardType)

public string GetSelectedCardTypeName(string cardValue)

public string GetSelectedMeasurementValue(string cardType, string measurement)

public int GetNumberOfExpectedMeasureState(string cardType)

public string GetEOF()

public string GetMeasureOn()

public string GetServiceOn()

public string GetMeasureOff()

public string GetConnect()

public string GetDisConnect()

public string GetRun()

public string GetResponseTranslate(string command, string data, string eof)

public string GetResponseData(string data)

public string GetResponseEOF(string data)

public bool ContainTheRespone(string input)

public bool IsCommonIncluded(string cardType)

public string GetDefaultName()

public int GetDefaultTimeOutValue()

public bool GetValidator(string data)

public string GetCurrentMeasurement(string cardType,int data)

private IEnumerable<String> GetMeasureListByCardTypeWithoutAuto(string cardType)

//A XML fájlokat olvassa és deszerializálja a Modelben felépített osztály.

public static class XmlProcessor

public static RootObject GetXmlRootObjectCommands()

public static RootObjectResponse GetXmlRootObjectResponse()

Model

public class Answer

public class Card

public class DefaultValues

public class Measure

public class NotifyViewModel : INotifyPropertyChanged

public void OnPropertyChanged(string propertyName)

public class Record

public class RootObject

public class RootObjectResponse

public enum UIElementStateVariations { ConnectBeforeClick, ConnectAfterClick, DisConnectClick, DisConnectBase, CardAndMeasureSelected, MeasureOffClick, MeasureOnAfterClick }

Resource

commandsTable.xml

responsesTable.xml

serviceCommandsTable.xml

View

MainWindow.xaml

ViewModel

//A parancsokat tartalmazza amelyek eseményeken keresztűl hívódnak meg a View felől.

public class MainViewModel : NotifyViewModel

private void MeasureTypeComboBoxChanged()

private void UpdateTimeUI()

private void ConnectToDevice()

private void ReadingSerialState()

private void DisConnect()

private void ConfigureDevice()

private void SendMeasureOn()

private void SendMeasureOff()

private void SendRun()

private void LoopMessagesArrayToSend()

private void DisConfigureDevice()

private void SendData(byte data)

private void PopUpQuestionbox()

private void TopMessage(string header, string text)

private void TimeOutValidator(TimeOutValidatorStates tovs)

private void TaskTimeOutWatcher()

private void DataRecieved(object sender, SerialDataReceivedEventArgs e)

private void WasItDisconnect()

private void SaveReport()

private void FolderDialog()