Daueraufträge

* Die [.Net Konventionen](https://docs.microsoft.com/de-at/dotnet/standard/design-guidelines/) auswendig lernen
* Die [C# Schlüsselwörter](https://docs.microsoft.com/de-at/dotnet/csharp/language-reference/keywords/) auswendig lernen
  + Teilgruppe Modifizierer
  + Teilgruppe Alias auf Datentypen
  + …
* Die [Operatoren](https://docs.microsoft.com/de-at/dotnet/csharp/language-reference/operators/) auswendig lernen
* Die [Datentypen](https://docs.microsoft.com/de-at/dotnet/csharp/language-reference/builtin-types/built-in-types) lernen

Einladung für zu Hause

Meisinger Klemens  
WIFI OÖ GmbH

3882 C# Teil 1 (Komponenten und Windows Forms)

3881 Komplettausbildung

**3883 C# Teil 2 (Windows Presentation Foundation, REST Schnittstelle, Datenbank und MVVM)**

3884 C# Projekt inkl. Prüfung (2er Gruppe eine eigene Aufgabenstellung)

Tipp für diejenigen, die die Prüfung machen:

* Ein „Kurstagebuch“, ein „Wissenstransferbuch“ schreiben
* In diesem in eigenen Worten die Inhalte von  
  einer Sitzung wiederholen
  + Dabei wird klar, was nicht klar ist
  + Das als „Frage“ notieren
  + In der nächsten Sitzung am Anfang stellen
* Beim Programmieren ist’s zu wenig,  
  eine Woche vor der Prüfung mit dem „Lernen zu beginnen“. Das Lernen für die Prüfung beginnt jetzt.

Termine

1. Di, 26.01.2021 17:30
2. [Do, 28.01.2021 17:30](#Jan28)
3. [Di, 02.02.2021 17:30](#Feb02)
4. [Do, 04.02.2021 17:30](#Feb04)
5. [Di, 09.02.2021 17:30](#Feb09)
6. [Do, 11.02.2021 17:30](#Feb11)
7. [Di, 23.02.2021 17:30](#Feb23)
8. Do, 25.02.2021 17:30
9. Di, 02.03.2021 17:30
10. Do, 04.03.2021 17:30
11. Di, 09.03.2021 17:30
12. Do, 11.03.2021 17:30
13. Di, 16.03.2021 17:30
14. Do, 18.03.2021 17:30
15. Di, 23.03.2021 17:30
16. Do, 25.03.2021 17:30
17. Di, 06.04.2021 17:30
18. Do, 08.04.2021 17:30

Zeitplan

17:30 – 18:50

19:00 – 20:20

20:30 – 21:35

Teilnehmer

**Bröll Michael**bin dabei, in Zukunft als Programmierer zu arbeiten…

**Eckertsberger Gerald**war in der Einführung; möchte die Anwendungsentwicklung vertieft sehen, …

Frage

* ***Basisbeispiel für die Rekursion ist „n!“***
  + Das faktorielle Produkt (Fakultät)   
    zur natürlichen Zahl n
  + 0! = 1
  + 1! = 0! \* 1 = 1
  + 2! = 1! \* 2 = 2
  + 3! = 2! \* 3 = 6
  + …
  + n! = (n-1)! \* n
* ***Wie kommt eine Anwendung   
  auf einen Benutzer-Rechner?***
  + „XCopy“ Verfahren, d.h. alles, was benötigt wird,  
    befindet sich im bin\release Ordner
  + Man erstellt ein eigenes „Installationspaket“
    - Den „Verpackungs- und Weitergabeassistent“  
      aus früheren Visual Studio Anwendungen gibt es nicht mehr
    - Heute in den Eigenschaften des Projekts  
      unter „Veröffentlichen“
    - Einen Installationsprogramm Assistenten kaufen
  + Wichtig  
    Falls die Anwendung „öffentlich“ wird, d.h. mit UWP  
    z. B. in den Windows Store oder das Installationspaket verkauft wird, muss das Projekt unbedingt elektronisch zertifiziert sein, d.h. man muss sich ein Zertifikat kaufen.

**Ertl Lukas**war in der Einführung; möchte nach der Reihe Anwendungen erstellen können, …

**Hannouf Samuel**Kenntnisse aus der Schule; möchte mich verbessern, Code sauber schreiben, …

**Humer Peter**Kenntnisse aus der Schule; möchte mein Grundlagenwissen verbessern, Neues lernen, …

**Kasper Daniel**habe im Vorjahr die Einführung und Teil 1 gemacht – jetzt geht’s weiter…

**Lindenhofer Christian**war in der Einführung; möchte auch mein Wissen vertiefen und bin gespannt, wie’s weiter geht, …

**Minehuber Christoph**Kenntnisse aus der Firma; Wissen vertiefen und erweitern, …

Frage

* ***Fehler im FensterManager vom WIFI. Wird eine Fenster auf einem Schirm geschlossen, der unterwegs, bei einem Notebook, nicht zur Verfügung steht, öffnet das Anwendungsfenster auf einem nicht existierenden Schirm und ist dadurch nicht sichtbar. Wie kann das Problem gelöst werden?***
  + .Net hat hier nix
  + Das Betriebssystem hätte das
  + Wie kann ich Betriebssysteme in C# realisieren?  
      
    System.Runtime.Interopservices.DllImportAttribute
* ***Was wird aktuell für Multithreading benutzt?***
  + Das hochkomplex
  + Führt zu zahlreichen Abstürzen
  + Microsoft sagt, das Original nicht benutzen
  + Heute: Aufgaben-orientiertes-asynchrones Programmieren: [TAP](https://docs.microsoft.com/de-de/dotnet/standard/asynchronous-programming-patterns/)
  + Für dieses TAP hat Microsoft einen   
    neuen Namespace System.Threading.Tasks   
    direkt in der Kernbibliothek eingeführt.  
    Damit steht TAP überall bereit.

**Plaimer Markus**war in der Einführung; Anwendungsentwicklung vertiefen, …

Zum Inhalt vom 3883

* Aktuelles Projekt vor einer Sitzung  
  ist gezippt auf Laufwerk Q:
* Das Zip auf die lokale Maschine kopieren  
  und den enthaltenen Ordner unter   
  Dokumente/Visual Studio/Projects

Projektmappe WIFI.CS.2021

* Mit den bisherigen Projekten 3880 (Einführung) und 3881(Teil 1)

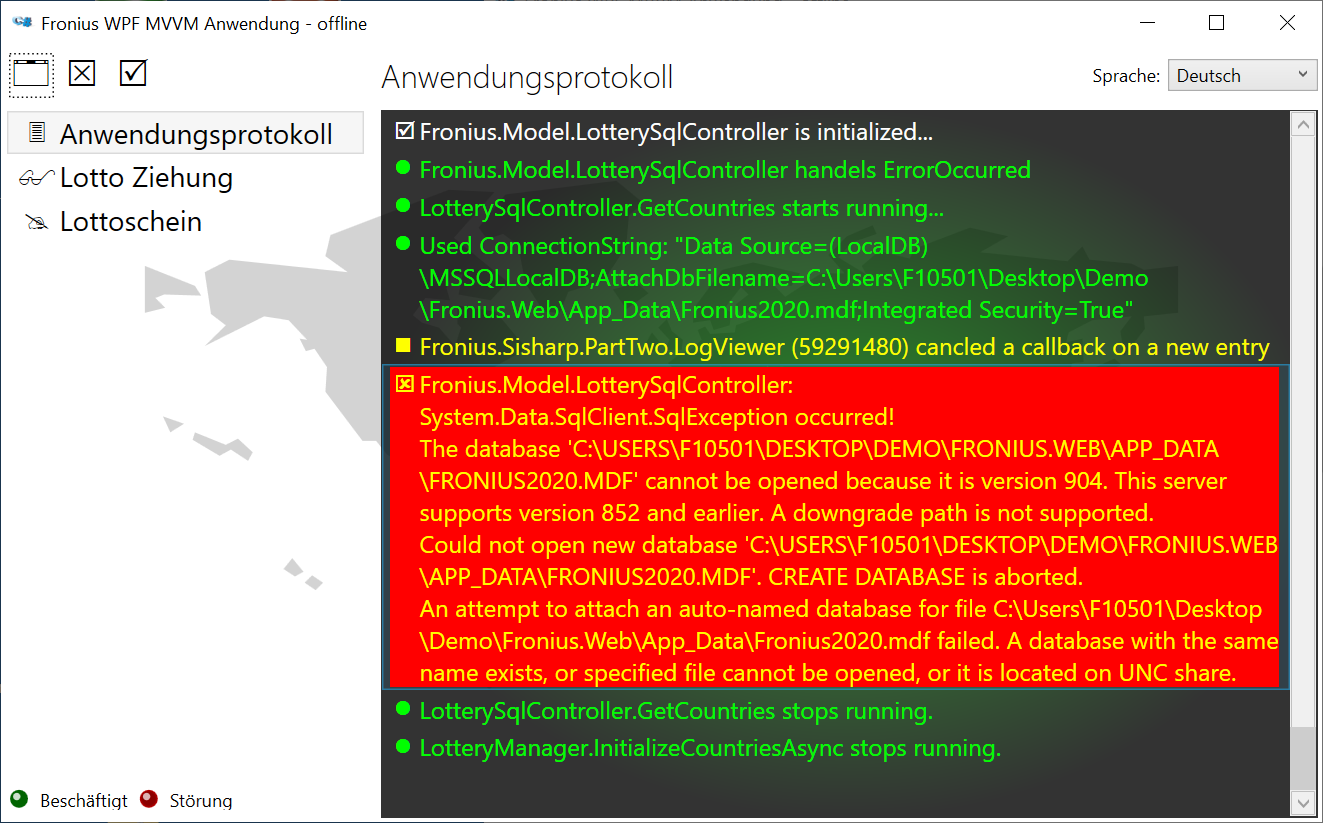
Wer eine Maschine hat, wo bereits gearbeitet wurde (Hr. Bröll und Hr. Kasper), Studio zurücksetzen (Menü Extras) und Menü Fenster / Fensterlayout zurücksetzen

* Immer offen  
    
  Der Objektkatalog (Menü Ansicht)

Worum geht’s?

* Wir sollen eine grafische Client Anwendung machen,  
  mit der man sich die Zahlen von Lottoziehungen   
  ansehen kann

Skizze



Wir wissen…

* Eine Anwendung, ein Projekt, beginnt mit der  
  Analyse. Heute objektorientiert und mit der  
  UML beschrieben (in der Einführung und im Teil 1)
* In diesem Teil 2 vom Trainer durchgeführt
* Wird wieder im Projekttraining spannen,  
  weil die Analyse abzugeben ist

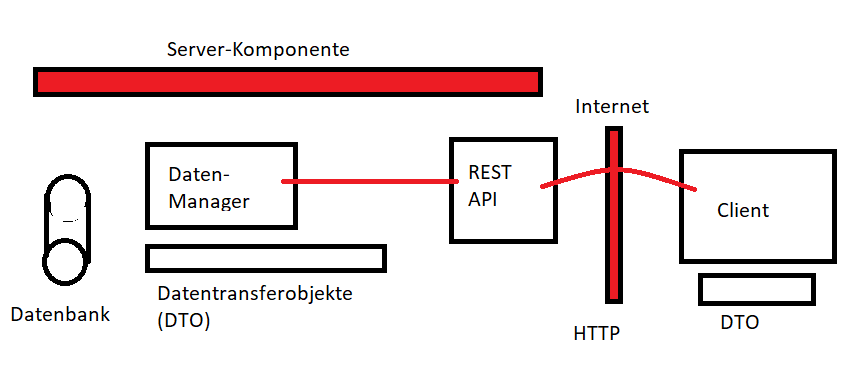
Außerdem wissen wir…

* Eine Anwendung wird auf Komponenten aufgeteilt
* Die bisherige Komponente „WIFI.Anwendung“

Hinweis

* Damit das Visual Studio Klassendiagramme  
  darstellen kann, im Visual Studio Installer  
  unter Einzelkomponenten den Klassen-Designer aktivieren
* Was brauchen wir?
  + .Net Desktopentwicklung
  + ASP .Net und Webentwicklung  
    (Hier wird auch der SQL Server LocalDB installiert)  
      
    und
  + die Einzelkomponente Klassen-Designer

Zur WIFI Client / Server Anwendung:



* es muss eine Client / Server Anwendung werden
* Dazu wird benötigt
  + Eine (1) Komponente, in der  
    nur das Aussehen der Datenobjekte  
    enthalten ist. Diese Komponente  
    ist auf dem Server und beim Client  
    (damit dieser das nicht neu machen muss)
  + Eine (2) Komponente auf dem Server,  
    der die Lottodaten / Lottoscheine  
    in der Datenbank verwaltet
  + Eine Client-Anwendung (3)
  + Zur Kommunikation eine  
    (4) Schnittstelle, die überall angesprochen  
    werden kann, also HTTP.  
    So eine Schnittstelle wird als REST API bezeichnet
* Wir benötigten vier neue Komponenten
* Gespeichert ist so eine Komponente als Assembly
* Beim Benennen die Konventionen einhalten

Also

* Erste Schritte in einem neuen Projekt,  
  in einer neuen Komponente
  + Alle Dateien anzeigen
  + Assembly-Info
  + Einstellungen zum Erstellen,  
    z. B. Xml-Dokumentationsdatei

Zum Schwierigsten:

* Was benutzen wir als Client?
  + Microsoft Studio Code  
    (nicht verwechseln mit Visual Studio)  
    ist eine „reine“ Javascript Anwendung,  
    d.h. in Wirklichkeit ein „Chromium“,  
    der nicht mehr als Chromium erkennbar ist.  
      
    Als „Framework“ wird „Electron“ vom MIT benutzt  
      
    Vorteil:  
    Diese Anwendung sieht unter Windows  
    genauso aus wie unter Linux (und der Mac ist nix anders)  
      
    Aufpassen, „Javascript“ auf keinen Fall   
    mit „Java“ von Oracle verwechseln
  + Wir wollen eine Oberfläche mit C#
    - Konsole (war in der Einführung)
    - Windows Forms  
      (heute nicht mehr scharf,  
      wenn man ein anderes Zoom als 100 % verwendet)
    - **Windows Presentation Foundation (WPF**)  
      (läuft nur unter Windows)
    - [Xamarin](https://de.wikipedia.org/wiki/Mono_(Software)) (mittlerweile von Microsoft gekauft)  
      (gibt’s für Windows und Linux (und damit auch am Max)
    - Web Oberfläche (ASPX)
* Wie machen wir die [REST](https://de.wikipedia.org/wiki/Representational_State_Transfer) Schnittstelle
  + Wir könnten das mit einer Konsolenanwendung „manuell“ machen, z. B. Python, wir interpretiert und zahlreiche Bibliotheken für den wissenschaftlichen und technischen Bereich, z. B. Machine Learning,  
    z. B. PGAdminIV damit realisiert. PGAdminIV wird zum Verwalten einer der leistungsfähigsten Datenbanksysteme benutzt. Das ist Postgresql (gespr. „Post-Gres-Ql“. Zum Beispiel von der Linz AG eingesetzt
  + Wir wollen ASPX verwenden.  
    Hier hat Microsoft Visual Studio einen Assistenten,  
    der eine REST Webanwendung konfiguriert

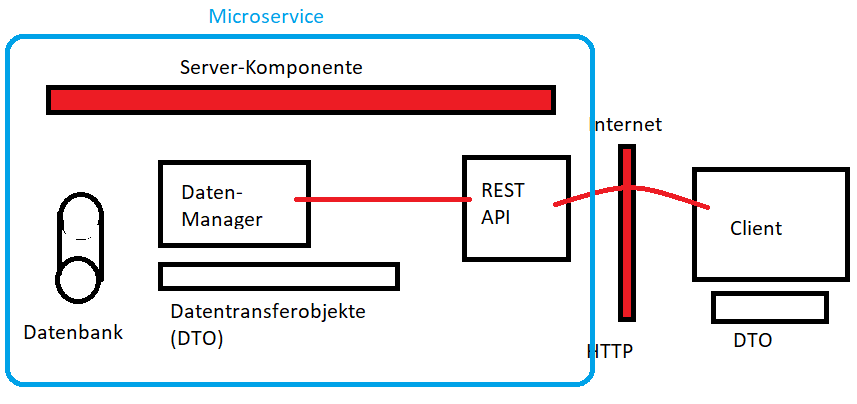
Inventur

|  |  |
| --- | --- |
| Assembly | Beschreibung |
| WIFI.Lotto.Daten (Neu) | Enthält die Datentransferobjekte. Diese Assembly ist am Server und beim Client. Hier ist keine Logik. Das ist getrennt, damit das beim Client nicht extra getippt werden muss |
| WIFI.Lotto (Neu) | Stellt das „Modell“ für die Lotto Demoanwendung bereit |
| WIFI.CS.Teil2 (Neu) | Die Client Anwendung. Demonstriert wie Server Komponenten benutzt und visualisiert werden |
| WIFI.Gateway (Neu) | Stellt ein REST API zum Veröffentlichen vom WIFI.Lotto Modell bereit |
| WIFI.Anwendung | Stellt das Basis-Firmenframework für eine WIFI Lösung bereit. Hier gibt es keine Datenbankunterstützung |
| WIFI.Anwendung.Datenbank (NEU) | Erweitert das Firmenframework um Datenbank-Funktionalität |

Hinweis:

* Ein ASP .Net Projekt holt zahlreiche   
  Assemblies aus dem Internet
* Das sind so genannte Nugets
* Diese können über das Kontextmenü  
  von „Verweise“ verwaltet werden
* Jetzt ist die Größe des Projektordners explodiert
* Die aus dem Internet geholten Nuget  
  befinden sich Projektordner unter „Packages“
* Im Projektordner WIFI.CS.2021  
  gibt es eine Batch-Datei „BinDatenEntfernen.bak“,  
  mit der die Binärdaten automatisch gelöscht  
  werden können. Vorher von „\*.bak“ auf „\*.cmd“ ändern.

Wie nennt man diese Software – Architektur?



* Das sind die so genannten [Microservices](https://de.wikipedia.org/wiki/Microservices)
* Das „Schlagwort“ in der Softwareentwicklung
* Microsoft hat auch [kostenlose E-Books](https://docs.microsoft.com/en-us/dotnet/architecture/microservices/) in dem Bereich

Was versteht man unter Microservice?

* Kleine kompakte Lösungen für Teilaufgaben   
  von etwas Großem
* z. B. unser WIFI.Lotto
* Die Kommunikation mit so einer Lösung  
  läuft über eine REST Schnittstelle
* Diese REST Schnittstellen können  
  in einem „Gateway“ mehrere Microservices sammeln
* Vorteil
  + Man kann eine Anwendung  
    auf mehrere Computer verteilen
  + Solche „Computer“ können auch „virtuell“ sein
  + Problem  
      
    Bei „virtuellen Computern“ ist  
    immer das ganze Betriebssystem enthalten  
      
    Aktueller Trend  
    Nicht einen ganzen Computer virtualisieren,  
    nur eine Anwendung, ein Microservice  
      
    Hier gibt es eine Drittanbieter Software,  
    die darauf spezialisiert ist…  
      
    Der [Docker](https://www.docker.com/)

Damit kann mit dem Umsetzen begonnen werden

Komplett neu:

* Webdienst, das WIFI Gateway

Wo beginnt eine ASP .Net Anwendung zu laufen?

* Im global.asax

Was benötigt eine ASP .Net Anwendung?

* Das kommt auf das Framework an
  + ASP .Net Core  
    läuft unter Linux, z. B. auch unter dem Apachen
  + ASP .Net Framework  
      
    Das benötigt den Internet Information Server (IIS)  
    Das ist das Problem der WIFI Client / Server Anwendung, wenn Teilnehmer die Anwendung „einfach so probieren“ wollen.  
      
    Weil ohne IIS und gehostetem WIFI.Gateway  
    läuft die Anwendung nicht.  
      
    Warum läuft’s im Visual Studio?  
      
    Weil das hat „IIS Express“. Rechts unten  
    in den System-Icons der Taskleiste  
      
    Wie kommt man zum IIS?
    - Der ist im Lieferumfang  
      vom Windows Server enthalten  
      (und Lizenzgebühren)
    - Im Windows Pro (nicht im Home)
* Wunsch an die Client Software
  + Sie muss auch „Offline“ funktionieren
  + D.h. sie muss konfiguriert werden  
    können, dass ohne WIFI Gateway  
    mit WIFI.Lotto „offline“ gearbeitet werden kann
  + Hier soll auf WIFI.Lotto KEIN Verweis gemacht werden
  + WIFI.Lotto Assembly muss dynamisch nachgeladen werden (z. B. als Add-In)
* Was braucht deshalb unsere Anwendung Teil2
  + Mind. zwei Konfigurationseinstellungen
    - UrlGatewayAPI
    - OfflineModus
* Wie kommt man zur URL vom Gateway?
  + Den Teil2 zum Startprojekt
  + Beim Starten muss aber  
    auch das Gateway gestartet werden
  + Das muss extra konfiguriert werden
    - WIFI.Gateway markiert
    - In den Eigenschaften  
        
      „Beim Debuggen starten: True“
  + Im IIS Express kann die Entwicklungs-URL  
    nachgesehen und in der Konfiguration vom Teil 2 eingetragen werden
* Wo müssen REST Api Controller  
  bei ASP .Net hinterlegt werden?
  + Im Projektunterordner „Controllers“
  + Der Name von so einem Controller  
    muss auf „Controller“ enden
  + z. B. „LottoController“  
    mit https://serverIPorDNS/api/lotto  
    wird das benutzt
* Dauert noch eine Weile, bis wir dort sind

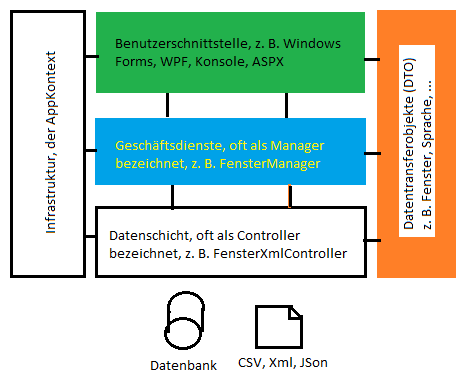
Zurück zum Teil 2

Eine WPF Anwendung

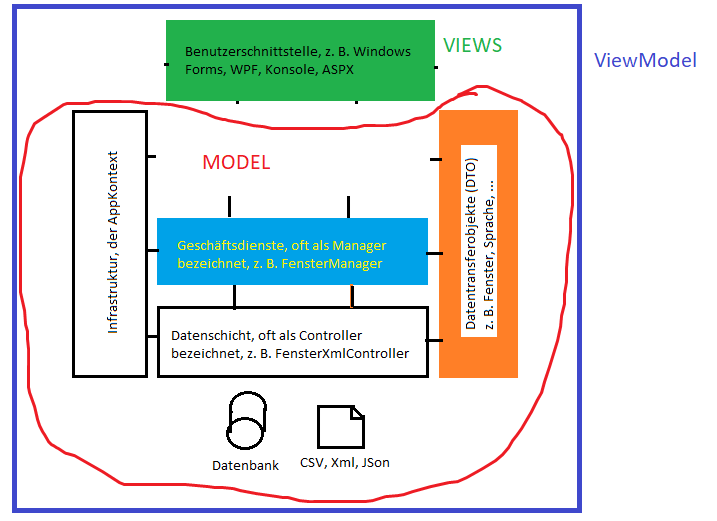
* Wo beginnt eine .Net Anwendung zu laufen?  
    
  Im „Main()“
* Bei WPF hat Microsoft das Main() versteckt
  + Mit dem sind wir totunglücklich
  + Wir wollen unser Main zurück,
    - das benötigen wir für  
      unsere Infrastruktur
    - den AppKontext
* damit das Main() nicht von einem Tool generiert wird,  
  muss bei der automatisch erstellten Datei „App.xaml“  
  in den Eigenschaften der Buildvorgang  
  von ApplicationDefinition auf „Page“ umgestellt werden
  + „Page“ ist wichtig, damit  
    später die Ressourcen funktionieren
* Wo macht man das eigene Main()
  + Am besten im Code behind  
    vom App.xaml
* Von wo nehmen wir die Infrastruktur?
  + Aus unserem Firmenframework

Zum Aufbau einer Anwendung:

* Wir kennen das Mehrschichtenmodell



* Dieses Mehrschichtmodell wird  
  durch ein Programmiermuster erweitert
  + Ziel  
      
    In der Benutzerschnittstelle  
    „darf kein Code sein“  
      
    Wird als „Views“ bezeichnet  
      
    Was ist dazu notwendig?  
      
    Eine (perfekte) Datenbindung
    - Windows Forms ist  
      hier schwach
    - Wer kann das?  
      bei WPF das  
        
      Xaml
    - Steht für  
      „eXtensible Application Markup Language“
    - Eine Erweiterung von Xml,  
      das kennen wir aus dem Teil 1  
      von unseren „Sprachen.xml“,  
      wo auch Klassen initialisiert werden



* Wegen der Datenbindung werden die „Views“  
  losgelöst vom Rest, dem „Model“
* Zusammengeführt, „gesteuert“ werden  
  die Views und die Models vom ViewModel

Das ganze Programmiermuster heißt …

M-V-VM  
  
 „[Model-View-ViewModel](https://de.wikipedia.org/wiki/Model_View_ViewModel#:~:text=Model%20View%20ViewModel%20(MVVM)%20ist,Logik%20der%20Benutzerschnittstelle%20(UI).)“

WPF Anwendungen sind für dieses MVVM ausgelegt,  
z. B. das Visual Studio

Deshalb gibt es jetzt im Teil 2 neue Ordner „Models“ und „ViewModels“

* Was macht der Trainer nicht?
  + Es gibt Projekte, wo auch   
    ein Ordner „Views“ vorhanden ist
  + Diese „Views“ liegen beim  
    Trainer aber auf der Hauptebene

Erster Schritt:

* Die WIFI Firmensprache ist Deutsch
* Deshalb sind auch die WIFI Klassen deutsch
* Das „Mainwindow.xaml“ löschen  
  (auch in App.xaml“ entfernen)
* Ein neues WPF Fenster über das Kontextmenü  
  hinzufügen und „Hauptfenster.xaml“ benennen

Erste Schritte in Xaml

* Das Format ist Xml
* Als Xml-Elemente werden aber  
  auch Klassen erkannt, z. B. „<Window> </Window>
* In einem Xml-Element die Attribute,  
  z. B. „Title“

Wie im Teil 1 muss unsere Oberfläche zwei Sprachen  
unterstützen

* Eine Sprache, die benutzt wird,  
  wenn’s keine Übersetzung gibt…  
    
  Das ist heute Englisch
* Für uns eine deutsche Lokalisierung
* Diese Texte kommen aus Ressourcen
  + Im Teil 1 wurde die Standard – Ressource  
    „Properties.Resources“ benutzt
  + Jetzt  
      
    Es sind auch eigene Ressource-Dateien erlaubt,  
    z. B. „Texte.resx“
* AUFPASSEN
  + Im MVVM werden Oberflächen „gebunden“
  + Gebunden werden können  
    NUR ÖFFENTLICHE EIGENSCHAFTEN
  + Der Ressourcen-Assistent ist hinterfotzig  
      
    Im Ressourcen-Editor oben auf   
    „Zugriffsmodifizierer: Public“ umstellen
* Das Lokalisieren ist im .Net Framework  
  tief verankert
  + System.Globalization
  + CultureInfo
  + Dort gibt’s einen Hyperlink im Docs  
    zu einer [Liste](https://docs.microsoft.com/de-de/openspecs/windows_protocols/ms-lcid/a9eac961-e77d-41a6-90a5-ce1a8b0cdb9c) aller unterstützten Sprachen,  
    z. B. „en“ für Englisch, „de“ für Deutsch  
    oder „de-AT“ für Österreich, …

Erstes Beispiel zur Datenbindung

* Im Hauptfenster.xaml soll der Titel  
  an die Ressource Texte.AppTitel gebunden werden

Es gibt drei Hauptbindungen

* An das ViewModel  
    
  {**Binding** EigenschaftVomViewModel}
* An eine Xaml-Ressource, definiert  
  z. B. in „App.xaml“  
    
  {**DynamicResource** NameDerRessource}
* An eine .Net Ressource, z. B. Texte.resx  
  Hier werden vom Studio  
  statische Eigenschaften in einem  
  Namespace Properties implementiert.  
  Zum Binden an diese…  
    
  {**x:Static** nn:Texte.AppTitel}

Worum handelt es sich bei „x:“ oder „nn:“?

* Mit Xaml können, im Gegensatz zu Xml,  
  auch Klassen als Objekte definiert werden
* Wie findet Xaml die Möglichkeiten?
  + Über Xml-Namespaces
  + Diese sind in einer Xaml Datei  
    immer oben

28. Jänner 2021

Unterschied in den .Net Namespaces

* Windows Forms (Teil 1): System.Windows.Forms
* WPF (Teil 2): System.Windows

Erster Auftrag im MVVM Projekt:

* Ein ViewModel schaffen,  
  damit im Teil 2 das Xaml Fenster sichtbar wird

Problem

* Wir benötigen überall unsere Infrastruktur
* D.h. wir benötigten eine Klasse, die  
  von unserem Anwendungsobjekt erbt

Lösung:

* Verweis auf das Firmenframework WIFI.Anwendung.Datenbank

Warum?

* In WIFI.Anwendung gibt es AppObjekt
* Hier fehlen aber Sachen, die für  
  die MVVM Datenbankanwendung notwendig sind
* Wir wollen WIFI.Anwendung aus dem Teil 1  
  nicht mehr ändern (wurde versprochen)
* In WIFI.Anwendung.Datenbank wird  
  deshalb ein ViewModelAppObjekt erstellt
* WIFI.Anwendung.Datenbank  
  benötigt auch WIFI.Anwendung, also Verweis

Im Main() der MVVM wird das eigene ViewModel Objekt initialisiert…

1. Aus dem Firmenframework Teil 1 WIFI.Anwendung  
   die Infrastruktur AppKontext initialisieren
2. Die WPF Anwendung initialisieren,  
   damit System.Windows.Application.Current  
   einen Wert bekommt
3. Über die Infrastruktur das Haupt-ViewModel produzieren,  
   damit in diesem die Infrastruktur ist
4. Dem ViewModel mitteilen, welche View benutzt werden soll
5. Die (Benutzer-)Konfiguration der Anwendung speichern
6. Die eigene Infrastruktur herunterfahren,  
   z. B. vom Fenstermanager die Fenster speichern

Was ist der Vorteil von MVVM?

* Erinnerung an den Teil 1 mit Windows Forms,  
  z. B. PrintPreviewDialog
  + Es gibt nur Einfachvererbung
  + Der PrintPreviewDialog  
    ist eine Erweiterung von Windows.Forms.Form
  + Hier gibt es unsere Infrastruktur nicht,  
    also kein FensterManager zum  
    Speichern und Wiederherstellen der Position
  + Deshalb (war Hausübung) eine  
    neue Klasse PrintPreviewDialog,  
    wo die eigne Infrastruktur und das  
    Speichern und Wiederherstellen  
    der Fensterposition getippt wurde  
    (WIFI.Windows.Forms)
* Im MVVM werden die System.Windows Klassen  
  nicht mit eigenem Code ausgestattet
* Der „eigene Code“, d.h. die ViewModels  
  werden „nur gebunden“
* Welche Basisklassen die ViewModels haben,  
  ist egal

Wichtig

* Wir wird eine View und das ViewModel  
  verbunden, sodass die Datenbindung funktioniert?  
    
  Indem bei der View die …  
    
  DataContext  
    
  … Eigenschaft eingestellt wird.

Übrigens:

* Möchte jemand die Dokumentation zur Idee  
  der WIFI MVVM Anwendung studieren,  
  im Docs das Kapitel zur WPF Datenbindung lesen  
  + Hauptdokumentation für WPF  
      
    [Windows Presentation Foundation - WPF .NET Framework | Microsoft Docs](https://docs.microsoft.com/de-de/dotnet/desktop/wpf/?view=netframeworkdesktop-4.8)
  + Teilkapitel zur Datenbindung  
    (von hier hat der Trainer die Idee  
    für die Schulungsanwendung)  
      
    [Übersicht über Datenbindung - WPF .NET Framework | Microsoft Docs](https://docs.microsoft.com/de-de/dotnet/desktop/wpf/data/data-binding-overview?view=netframeworkdesktop-4.8)

Über das Main() und VM.Starten haben wir also  
jetzt die Kontrolle über die Oberfläche.

Erster Wunsch:

* Die Anwendung soll sich die Fensterposition merken

Änderung zu Windows Forms

* Es gibt in WPF kein Load-Ereignis,  
  d.h. wir können nicht wie im Teil 1  
  das OnLoaded überschreiben
* Ohne MVVM ist das aktuelle Problem  
  nahezu unlösbar
* Mit dem MVVM kostet es uns ein müdes Lächeln
* Wir haben die Kontrolle über das „new Window“
* Hier können wir die Fensterposition wiederherstellen
* Damit das „Starten“ nicht zu umfangreich wird,  
  auslagern in eine private Methode

Erinnerung

* .Net
  + Zwei Datentyp-Familien
    - Werttypen  
      („struct“)  
      Hier hat man die Information direkt.  
      Die Standardinitialisierung ist die Zahl 0  
      Werttypen können nicht null („keine Speicheradresse“) sein. Manchmal nervig.  
      z. B. „int“ für Sytem.Int32
    - Verweistypen  
      („class“)  
      Hier hat man nur die Speicheradresse  
      auf ein Objekt. Standardinitialisierung null
  + Praktisch sein .Net 2  
      
    Nullable Werttypen  
    In der Syntax ein Suffix „?“,  
    z. B. „int?“, ein nullable Werttyp

Problem

* Bug im Firmenframework WIFI.Anwendung
* Der Datenpfad ermittelt über  
  die Assembly Information den Speicherort
* Es wird nicht geprüft, ob  
  unerlaubte Zeichen vorhanden sind

Wichtig

* Die Gefahr bei MVVM ist,  
  dass man „Speicherlöcher“ bekommt
* Nicht mehr benutzte Objekte, d.h.  
  Objekte, deren Speicheradresse in  
  keinen Variablen ist, werden vom  
  Garbage Collector entfernt
* Weil wir das ViewModel in  
  den DataContext der View stellen,  
  kann der Garbage Collector die View   
  nicht entfernen, weil das ViewModel  
  noch läuft
* Deshalb unbedingt beim Schließen  
  eines Fensters, den DataContext freigeben

Wo speichert der FensterManager die Daten?

* Eine Anwendung darf auf keinen Fall  
  in das Anwendungsverzeichnis schreiben
* Ungefragt darf nur in das Benutzerprofil gespeichert werden
* In diesem gibt es einen versteckten Ordner „AppData“
* Am schnellsten kommt man zu diesem,  
  wenn im Windows Explorer „%Appdata%“ eingeben wird
  + AppData wird in
    - Local („AppKontext.LokalerDatenpfad“)
    - Roaming („AppKontext.Datenpfad“)

unterschieden. In „Local“ kommen die Daten,  
die nur Computer bezogen sind. In „Roaming“  
kommen die Daten, die mit Server synchronisiert werden sollen

Am Ende der zweiten Sitzung

* Wir hätten gerne im Hauptfenster eine  
  Liste, wo die Sprachen ausgesucht werden können
  + Wo sind die Sprachen?
    - Im ViewModel
    - Dieses hat eine Eigenschaft AppKontext
    - Der AppKontext hat eine Eigenschaft Sprachen
    - Sprachen (ein Sprachenmanager) hat  
      eine Eigenschaft StandardListe
  + In MVVM wird nur „gebunden“
  + D.h. wir brauchen eine Liste gebunden   
    an AppKontext.Sprachen.StandardListe
* Wird mit Xaml umgesetzt

Wie wird in Xaml „positioniert“?

* Basispositionieren ist <Grid>  
    
  Dieses hat Zeilen und Spalten
* Neben dem Grid können Objekte  
  auch mit <Stackpanel> positioniert werden

Tipp beim Arbeiten mit Xaml

* Kommentieren
* Dazu den Xml Kommentar <!-- -->

Hinweis

* Die Größenangaben sind Pixel
* Ausnahme  
  Bei Schriftgrößen „Punkte“ verwenden.  
  Bei Schriften also nicht z. B. FontSize="12"  
  „12“ würde als Pixel interpretiert  
  Schriftgrößen sind aber „Punkte“ und  
  ein Punkt ist nicht ein Pixel  
  z. B. FontSize = "12pt"

Wichtig beim Xaml

* Selber tippen. Nicht mit der Maus irgendetwas verschieben
* Wird mit der Maus verschoben oder  
  werden Änderungen im Eigenschaften-Fenster  
  durchgeführt, steht als „nicht lesbar“ in einer Zeile
* UNBEDINGT sauber strukturieren
* Mit Zeilenumbrüchen arbeiten
* Sonst verliert man die Kontrolle über Xaml

Cool bei der Datenbindung:

* Es kann jetzt von einem Objekt auch  
  eine untergeordnete Eigenschaft benutzt
* D.h. es kann ein Zugriffspfad festgelegt werden

2. Februar 2021

Zum Aufwärmen:

* Damit das Hauptfenster nicht so leer ist,  
  eine Weltkarte positionieren

Lösung:

* Von wo nehmen?
  + z. B. aus einer Blickfangpunkt-Schrift,  
    weil ein Zeichen nicht viel Speicher benötigt
  + Fündig: Zeichentabelle vom Betriebssystem  
    in der Schrift „Webdings“
  + Das Zeichen 0xFB
* In Xaml umsetzen mit einem [Steuerelement](https://docs.microsoft.com/en-us/dotnet/desktop/wpf/controls/control-library?view=netframeworkdesktop-4.8),  
  z. B. Label im Hauptfenster.xaml

Hinweis

* In Xml haben viele Zeichen eine Bedeutung
* Früher oder später fällt jeder d’rüber,  
  dass ein Zeichen beim Tippen einfach nicht funktioniert

Lösung

* „[Benannte HTML-Zeichen](http://www-hera-b.desy.de/subgroup/computing/IT/www/selfhtml/html/referenz/zeichen.htm)“  
    
  z. B. für das Hexadezimale Zeichen 0xFB   
  der Weltkarte in HTML &#xFB;

Lästig in WPF

* Im Jahr 2000 hat sich Microsoft gerühmt,  
  zusammengeräumt zu haben,  
  d.h. bei allen Steuerelement ist die sichtbare Bezeichnung „Text“
* Bei WPF haben’s das wieder vergessen
  + Hier ist’s ständig anders
  + Einmal „Text“
  + Einmal „Header“
  + Einmal „Content“  
      
    der Trainer könnte durch die Decke gehen

Hinweis zu den Farben in WPF (in Xaml)

* Das können benannte Farben sein
* Man kann aber auch Hexadezimal mischen
  + Transparenz: FF nicht transparent (Alpha)
  + Rot
  + Grün
  + Blau

Nächster Wunsch

* Aktuell haben wir im Hauptfenster   
  unsere Sprachen in der Liste
* Es ist aber noch keine ausgewählt

Wichtig:

* Damit WPF Änderungen in einer Eigenschaft mitbekommt,  
  **muss** ein Ereignis ausgelöst werden
* Ereignis PropertyChanged

Lösung

* Wir wollen das Ereignis nicht in unserem  
  ViewModel Teil2 Anwendung
* Weil wir haben ja später mehrere ViewModels
* Das Ereignis gehört in eine Basisklasse
* Hier haben wir unsere WIFI.Anwendung.ViewModelAppObjekt  
    
  in der Assembly WIFI.Anwendung.Datenbank

Wichtig

* PropertyChanged alleine reicht nicht
* WPF prüft nicht die Existenz der Eigenschaft
* WPF prüft nur, ob das Interface vorhanden ist,  
  d.h. INotifyPropertyChanged muss implementiert sein

Lästig

* Wir haben einen Ereignisauslöser  
    
  OnPropertyChanged
* Diese benötigt den Namen der Eigenschaft,  
  z. B. this.OnPropertyChanged("AktuelleSprache")
* Das ist lästig
* Das OnPropertyChanged soll  
  den Namen vom Aufrufer benutzen

Lösung

* Erst seit ca. 3 Jahren möglich

Wie kann Code gesteuert werden?

* z. B. ist unser Main() das letzte Mal  
  beim ersten Aufruf abgestürzt
* Lösung war: ein Attribut, das STAThread

Mit so einem Attribut kann jetzt auch   
OnPropertyChanged gesteuert werden

* Seit .Net 4.5 im CompilerServices  
    
  CallerMemberNameAttribute
* Damit das Attribut funktioniert,  
  muss der Parameter vom OnPropertyChanged  
  „optional“ werden,  
  d.h. ein Standardwert muss angegeben werden

In der Basisklasse ViewModelAppObjekt wurde also PropertyChanged vorbereitet

Jetzt kann im Teil 2 ViewModel.Anwendung  
die Eigenschaft für das aktuelle Land implementiert werden.

Notwendig, damit die Liste der Sprachen im Hauptfenster voreingestellt ist

Für eine Eigenschaft im Normalfall ein Feld.

Dieses Mal wird vom AppKontext.Sprachen die Einstellung als Feld benutzt

Nachdem das ViewModel jetzt eine Eigenschaft AktuelleSprache hat, kann diese in der Liste vom Hauptfenster gebunden werden

Hinweis:

* Die aktuelle Sprache muss natürlich voreingestellt werden
* So wie im Teil 1 im Main()  
  die Sprache über eine Anwendungskonfiguration steuern
* Die vom Benutzer verwendete Sprache  
  in der Anwendungskonfiguration hinterlegen
* Ohne Standardwert, damit bei  
  einem neuen Benutzer die Sprache vom  
  Betriebssystem voreingestellt wird

Nächste Ausbaustufe

* Weil heute viele Benutzer mehrere Bildschirme verwenden, soll unsere Anwendung in mehreren Fenstern geöffnet werden können
* Deshalb wird beim Wechseln der Sprache  
  der Hinweis ausgegeben, dass die Anwendung zum Wechseln der Sprache neu gestartet werden muss

Zum Öffnen eines zusätzlichen Fensters benötigen wir ein Symbol, wollen wir in der Oberfläche ein Symbol

Problem:

* Wir müssen zahlreiche Attribute   
  beim Xaml Button festlegen…

Width="40"

Height="40"

Background="Transparent"

BorderThickness="0"

FontFamily="Wingdings 2"

FontSize="24pt"

* Das interessiert uns nicht
* Wir wollen das in der Anwendung   
  nur einmal machen

Lösung

* Voi cook in WPF
* Man kann, wie in einer Textverarbeitung  
  die Absatzformatvorlagen, überall „Stile“ definieren
* Dazu hat jedes WPF Steuerelement  
  die „Resources“ Eigenschaft
* Möchte man, dass solche Stile  
  anwendungsweit gelten, in „App.xaml“ definieren

Neues Problem

* In der Windows Forms vom Teil1 haben wir bei einer Schaltfläche „einfach im Code Behind“ einen Ereignis-Behandler angehängt
* Das ist jetzt in MVVM verpönt

Was machen wir?

* Naja, binden, aber was
* Es sind „nur öffentliche Eigenschaften“ erlaubt
* Im aktuellen Fall zum Öffnen eines neuen Fensters wird aber eine Methode benötigt
* An eine Methode KANN NICHT GEBUNDEN werden

Lösung

* Eine Eigenschaft machen, in  
  der eine Methode gekapselt ist
* Das ist in WPF mit ICommand vorgesehen  
  im Namespace System.Windows.Input der System Assembly
* Objekte, die ICommand implementieren,  
  können für die Bindung benutzt werden
* Die im Objekt gekapselten Methoden  
  werden über den Konstruktor beim Initialisieren  
  mitgegeben
* Im Training WIFI.Anwendung.Befehl

Hinweis:

* Oft rufen überladene Methode  
  eine andere Variante auf
* Wie sieht das beim Konstruktor aus?
  + Damit ein Konstruktor eine  
    andere Variante aufrufen kann,  
    muss nach der Schnittstelle  
    ein Doppelpunkt und „this“ benutzt werden

Damit das eigene Befehl-Objekt für die WPF Command Bindung fertig wird, fehlt noch das Ereignis „CanExecuteChanged“

Was muss das machen?

* Es muss mit dem WPF CommandManager gekoppelt werden

Das ist jetzt die letzte Möglichkeit im Rahmen der Ereignisse…

Problem

* ICommand ist in der System Assembly  
  einen Verweis auf diese gibt’s immer
* CommandManager ist aber  
  in der Assembly PresentationCore
* Dieser Verweis fehlt in WIFI.Anwendung.Datenbank
* Wird das jetzt gemacht,  
  sparen wir uns WIFI.Windows als Assembly  
  für die eigene WPF Unterstützung

Unser CanExecuteChange Ereignis muss  
mit dem CommandManager.RequerySuggested gekoppelt werden.

Dazu eigene [Ereignis Accessoren](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/events/how-to-implement-custom-event-accessors)

Zurück zum ViewModel Anwendung

* Der Befehl NeuesFenster soll eine Methode „NeuesFensterErstellen“ kapseln

Neben dem Anwendungssymbol „Neues Fenster“  
soll auch ein Symbol zum Schließen aller Fenster vorhanden sein. Dieser Befehl soll aber nur zulässig sein,  
wenn mehr als ein Fenster offen ist

Zur internen Methode im ViewModel „NeuesFensterErstellen“

* Dazu benötigen wir den Typ  
  vom Starten
* Der fehlt
* Wir benötigen eine interne Eigenschaft
* Hier wurde auf das Feld verzichtet  
  und eine implizite Eigenschaftendeklaration benutzt

4. Februar 2021

Zum Aufwärmen in der MVVM Anwendung

* Cool, seit Studio 2017 kann während der Laufzeit einer WPF Anwendung die Xaml Deklaration verändert werden
* z. B. ist die aktuelle Sprache nicht vertikal zentriert

Auftrag

* Hr. Minehuber  
    
  Der WIFI Fenster-Manager hat ein  
  Problem mit mehreren Bildschirmen.  
  Wenn ein Bildschirm, auf dem ein Fenster  
  beendet wurde, nicht vorhanden ist,  
  sieht man das Fenster nicht.  
    
  Gehört gelöst

Lösung:

* C# hat nix
* .Net hat auch nix

Lästig:

* Wir hoffen auf das Betriebssystem
* Getippt in C/C++
* Das Win32 API
* Das 1.000ende Methoden
* Zum Auslesen von Informationen  
  über den Bildschirm …  
    
  [GetSystemMetrics](https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getsystemmetrics)

Wie wird in C# eine „NICHT .NET Komponente“ benutzt?

1. Ohne Dokumentation geht nix

Diese Dokumentation ist bei Windows Aufrufen vorhanden – im Docs. Werden andere Komponenten von Drittanbietern benutzt, wird’s lästig. Man braucht die Dokumentation

1. .Net hat ein Attribut, mit dem angegeben wird, dass eine nicht .Net Methode benutzt wird  
     
   System.Runtime.InteropServices.DllImportAttribute

Service Pack für WIFI.Anwendung

* Änderungen in der FensterManager-Klasse

Damit wäre einmal ein Hauptproblem gelöst, wie wir in .Net zu Informationen kommen, die vom Betriebssystem und nicht von .Net bereitgestellt wird

Idee zur Problemlösung:

* Bisher wird im FensterManager  
  nur der Fenstername als Schlüssel benutzt
* Ab jetzt wollen wir zum Namen  
  auch noch ein Suffix anhängen, bei wie vielen Bildschirmen das war
* Weil Bildschirme heute dynamisch  
  an- und abgehängt werden können,  
  darf diese Information nicht gecachet werden
* Nachdem der FensterManager Klasse  
  eine interne Eigenschaft MonitorSchlüssel  
  beigebracht wurde, alle Methoden anpassen,  
  wo der Fenstername benutzt wird  
  und um den Schlüssel erweitern
  + Hinterlegen✓
  + Abrufen✓

Nachlesen: [Im Docs / C# Programmierhandbuch / Interoperabilität](https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/interop/)

Wichtiger Hinweis von Hr. Eckertsberger:

* Aufpassen, bisher wurde „nur .Net“ verwendet
* D.h. wird unser .Net Framework 4.8 Assembly  
  auf 5 (also Core) umgestellt und WPF auf Xamarin, läuft die Anwendung auch am Mac und unter Linux
* Mit der Kern Änderung DllImport("User32.dll") haben wir uns an Windows gebunden!!!

Nächster Punkt:

* In einer Objektorientierten Anwendung verliert man rasch die Kontrolle. Die Anwendung ist auf mehrere Komponenten und zahlreiche Objekt verteilt
* Damit man die Kontrolle behält,   
  muss so eine Anwendung UNBEDINGT mitprotokollieren

Auftrag

* Im Firmenframework WIFI.Anwendung Namespace ist in der WIFI.Anwendung.Datenbank Assembly ein **Protokolldienst** zu implementieren
* Dieser Protokolldienst muss  
  über die Infrastruktur, d.h. den AppKontext  
  veröffentlicht werden, damit er überall bereitsteht

Wie heißt’s so schön in den Apple Programmierrichtlinien?

* Man kann gar nicht zu viel protokollieren

In den Google / Android Richtlinien:

* Keine Programmzeile länger als 80 Zeichen!!!

Zum Protokolldienst:

* Wir wollen verschiedene Typen von Protokolleinträgen unterscheiden
* Dazu kennt .Net die „gelben Zettel“,  
  die Enumerations, in C# getippt „enum“

Problem:

* Wir haben eine Klasse ProtokollEintrag
* Wird beim Debuggen auf ein  
  Objekt ProtokollEintrag gezeigt,  
  liest man „WIFI.Anwendung.Daten.ProtokollEintrag“
* Das ist in einer Liste aber nervig,  
  weil man 100e Male das liest

Lösung:

* Was zeigt der Debugger an?
* Das Ergebnis von ToString() geerbt  
  von System.Object
* Damit das Debuggen komfortabler wird,  
  müssen wir beim ProtokollEintrag ToString() überschreiben

Dazu haben wir keine Lust

* Wir wollen eine eigene Basisklasse,  
  wo das ToString() überschrieben ist

Problem

* Wie soll das ToString() wissen,  
  welche Eigenschaften angezeigt werden soll?

Idee

* Wie weiß der Compiler, dass bei einem Parameter, der den Standardwert enthält, der Name vom Aufrufer eingesetzt werden soll?
  + Mit einem Attribut

Lösung:

* Wir implementieren eine Daten-Basisklasse  
  mit einem überschrieben ToString() und dieses ToString() soll über ein Attribut gesteuert werden

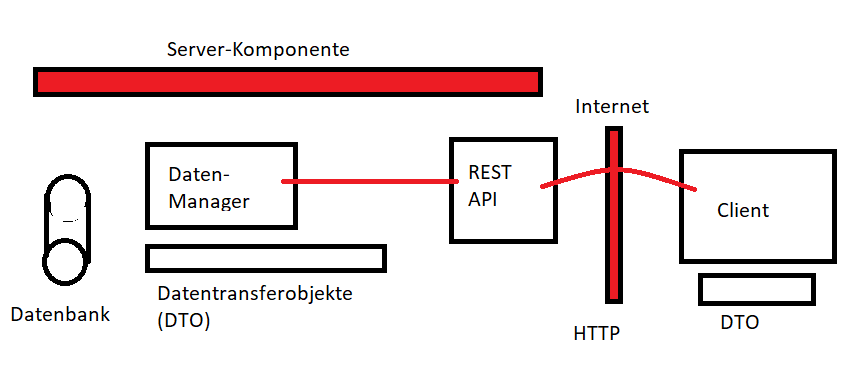
Wie wird ein Attribut implementiert?

* Es ist eine Klasse, deren  
  Namen auf „Attribute“ enden muss  
  und System.Attribute erweitert
* Es handelt sich um „Metadaten“
* Die Information wird „mitkompiliert“   
  und von interessierten Stellen benutzt werden

Zum Steuern der eigenen ToString() Methode  
soll das Attribut „InToString“ heißen

9. Februar 2021

Inventur der Anwendung  
(weil NIEMAND CODE ZEILEN AUSWENDIG WEISS!)



WIFI.CS.Teil2.exe (3)

WIFI.Lotto.Daten.dll (4)

WIFI.Lotto.Daten.dll (4)

WIFI.Lotto.dll (5)

WIFI.Gateway.exe (6)

Daneben werden die WIFI-Assemblies unterstützt durch…

* Firmenframework WIFI.Anwendung.dll (1)
* Firmenframework Erweiterung WIFI.Anwendung.Datenbank.dll (2)
* Im Kopf hat man die Klassendiagramme!

Hinweis zur Datenbindung in WPF

* An eine Ressource: {x:Static …}
* An eine WPF Ressource: {DynamicResource …}
* Im WPF, in der View an   
  das ViewModel im DataContext: {Binding …}  
    
  {Binding} kann auch benutzt werden,  
  um eine Eigenschaft an andere WPF Eigenschaften zu binden  
    
  NUR, wenn das WPF Objekt benannt ist

Weiter mit unserem ToString() in der DatenBasis,  
das mit einem Attribut gesteuert werden soll

Das Programm gibt es schon…

* Der Kommentar im ToString()

Es muss „nur noch getippt“ werden 😊

1. Wir benötigten dazu  
   die **Beschreibung** einer **Eigenschaft**,  
   die das **InToString**-Attribut hat  
     
   Das ist ein Substantiv, also Klasse,  
   z. B. InToStringEigenschaftBeschreibung
2. Weil wir mehrere solcher Eigenschaften  
   unter Umständen im Typ haben,  
   eine typsichere Liste dafür
3. Zum Sammeln der Listen InToStringEigenschaftBeschreibungen  
   ein „Wörterbuch“ verwenden,  
   eine „Hashtable“. s’Problem der Original Hashtable:  
   Sie ist nicht typsicher, d.h. als Schlüssel  
   wird „object“ und für Daten wird „object“ benutzt.  
   Wir wollen typsicher arbeiten, damit wir  
   nicht casten müssen. Dazu findet man  
   System.Collections.Generic.Dictionary<T1,T2>

Damit sind die Unterstützungsklassen deklariert.

WICHTIG

NICHT im Code „Variablen“ dieser Art deklarieren…

System.Collection.Generic.Dictionary<string,object> \_Wörterbuch = null;

UNBEDINGT so etwas mit einer eigenen Klasse typisieren. Die Lesbarkeit vom Code steigt dabei ungemein

* Ohne Probleme kann man in einer  
  Klasse private Unterstützungsklassen machen,  
  die später niemand sieht

1. Jetzt müssen wir sauber zwischen  
   „zur Klasse, zum Typ gehörend“   
   und „zum Objekt gehörend“ unterscheiden
   1. Zum Typ gehörend:  
        
      Eigenschaften mit InToStringAttribute
   2. Zum Objekt gehörend:  
        
      Die (aktuellen) Werte einer Eigenschaft  
        
      Das ERSTE IST „STATISCH“

Das „Wörterbuch“, der Cache zum Sammeln der Analyse muss statisch sein.

1. Problem beim Einsetzen der   
   aktuellen Werte in den ToString()  
   Ergebnistext
   1. Texte gehören unter Anführungszeichen
   2. Nummerische Daten nicht
   3. Null-Werte gesondert  
        
      d.h. wir müssen mind. 3 Ausgabevarianten  
      unterscheiden
   4. Diese Ausgabevarianten werden  
      in der Klasse InToStringEigenschaftBeschreibung  
      gekapselt

Frage Hr. Plaimer

* Beim Benutzen vom InToStringAttribute  
  werden die Eigenschaften in der Ausgabe  
  in der Reihenfolge des Codes aufgelistet
  + Im Grund wurscht
  + Weil dann kann man ja im Code  
    die Eigenschaften in der Reihenfolge  
    tippen, wie man’s im ToString braucht
  + Nichtsdestotrotz  
      
    Wie kann ich die Ausgabe der  
    InToStringAttribute steuern?
    - Wie bei den AssemblyInfos,  
      z. B. AssemblyTitleAttribute("Der Titeltext“)
    - D.h. man kann in Attributen  
      auch Daten verstecken
    - D.h. eine Eigenschaft z. B. „Index“  
      im InToStringAttribute
    - Das Ganze wird über den   
      Konstruktor initialisiert

11. Februar 2021

Beschwerde von Betatestern:

* Die Liste mit den Sprachen ist  
  nicht sortiert. Das ist fast so dämlich  
  wie bei Microsoft, z. B. beim  
  Lernen der Excel Tabellenfunktionen.  
    
  Problem: Das Original ist Englisch  
  und nach dieser Sprache sortiert.  
  Dann wird’s von einem Algorithmus  
  übersetzt und nicht neu sortiert  
    
  z. B. „English“ – „German“   
  oder „Deutsch“ – „Englisch“)  
    
  Muss vor dem Weitermachen gelöst werden

Wie?

* Wir haben bereits sortiert.  
  Wo? In der Einführung beim Lotto.BerechneQuicktipp  
    
  Dort war das aber auch kein Problem,  
  weil wir nur Zahlen im Array gehabt haben
* Jetzt haben wir aber ein  
  komplexes Objekt „Sprache“  
    
  Diese komplexe Objekt hat  
  einen Code und einen Namen
* Wie soll .Net jetzt wissen,  
  wie die Sprache sortiert werden soll,  
  nach „Code“ oder nach „Name“
* Wir wissen, nach „Name“

Lösung

* Voi, voi cool
* Ähnlich, wie in einer Datenbank

Wie heißt die strukturierte Abfragesprache auf eine Datenbank?

* „Structured Query Language“ (SQL)
* Standard seit Anfang 1970 von IBM entwickelt
* SQL hat zwei Befehlsfamilien
  + Daten Definitionssprache (DDL)
  + Daten Manipulationssprache (DML)
* Die DDL wird im eigenen Code eher selten benutzt,  
  dazu gibt’s grafische Werkzeuge
* Die DML wird aber ständig benötigt
  + Vier Hauptwörter  
    (jedes Wort hat dann unzählige Unterwörter)

1. Holen von Daten  
     
   [SELECT](https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql?view=sql-server-ver15)  
   (Das Microsoft Training dazu  
   dauert ganze 5 Tage)
2. Einfügen von Daten  
     
   INSERT
3. Ändern von Daten  
     
   UPDATE
4. Löschen von Daten  
     
   DELETE

* SQL wurde vom ANSI genormt
* Jeder Hersteller hat aber seinen „Dialekt“
* Der Dialekt bei Microsoft heißt  
    
  „[Transact-SQL](https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15)“, kurz T-SQL

Microsoft wollte, dass ähnlich wie SQL, auch in der .Net Programmierung auf Daten, die bereits im Arbeitsspeicher sind, zugegriffen werden kann…

Die Frage war, wie?

* In eine Basisklasse können wir   
  die Lösung nicht geben
* Es gibt nur Einfachvererbung  
  und die meisten Objekte, wo  
  das benötigt würde, wären keine  
  Erweiterung der Basisklasse mit der Lösung
* Was machen wir?  
    
  Wir benötigen eine Technik,  
  mit der wir Objekten „etwas unterjubeln“  
  können, ohne dass das aus dem Stammbaum kommt  
    
  Mit .Net 2 hat Microsoft Lösung  
  die Erweiterungsmethoden erfunden.  
  Diese Technik kennen wir, das sind  
  die Methoden, die im Symbol einen schwarzen Pfeil haben. Im Teil 1 realisiert.  
    
  Wie werden Erweiterungsmethoden aktiviert?  
  Mit der „using“ Direktive auf den Namespace,  
  in der sich die Klasse mit den Erweiterungen befindet.
* Nachdem die Erweiterungsmethoden  
  mit .Net 2 funktioniert haben, hat Microsoft  
    
  „die Sprachintegrierte Abfragesprache“  
    
  entwickelt in Form von Erweiterung, also  
  Methoden mit schwarzem Pfeil  
    
  „[Language Integrated Query](https://docs.microsoft.com/en-us/dotnet/csharp/linq/)“ oder „LINQ“  
    
  Seit 2008 läuft LINQ auch „parallel“,  
  d.h. LINQ zerlegt eine Aufgabe in Teilaufgaben  
  und lässt jeden Teil einem eigenen Thread laufen,  
  damit die Kerne des Prozessors ausgelastet sind.
* Wer SQL SELECT kennt, kann sofort mit LINQ arbeiten, weil’s im Grunde dasselbe ist
  + Unterschied  
      
    SQL  
      
    SELECT Feldliste FROM Quelle   
    und z. B. ORDER BY Feld  
      
    LINQ  
      
    FROM Lambda IN QUELLE   
    ORDERBY Lambda.Eigenschafdt  
    SELECT Lambda
  + Der Trainer „liebt“ LINQ

Zurück zum aktuellen Problem

* Wir müssen die komplexen   
  Sprache-Objekte sortieren
* Lösung: LINQ

Details zu den Sprachen

* Physikalischen sind diese  
  in einer Xml Datei in den Ressourcen
* Diese Ressourcen werden vom  
  Controller, dem XmlSprachenController gelesen
* Dem Controller ist die Reihe wurscht,  
  und muss im wurscht sein
* Dass die Sprachen sortiert sein sollen,  
  ist ein Problem von der 2. Ebene, den Managern,  
  in unserem Fall dem SprachenManager

Beispiel für LINQ:

Eine Abfrage noch ohne Daten:

from s in this.Controller.HoleStandardListe()

orderby s.Name

select s

Diese Abfragen müssen „materialisiert“ werden.  
Dazu ist der Ausdruck in Klammer zu setzen

Zum Materialisieren gibt’s zahlreiche Erweiterungsmethoden,  
z. B. falls mit „where“ gesucht würde, „First()“ bzw. „FirstOrDefault()“ oder „Max()“, „Min()“, „Sum()“, …

Falls mehrere Ergebnis geliefert werden, „ToArray()“ oder „ToList()“,…

Was hat LINQ mit C# zu tun?

* Nix, das ist eine .Net Möglichkeit

Nach diesem Wunsch der Beta-Tester zurück zum Protokoll:

* Aktuell haben wir EINEN Protokolleintrag
* Wir benötigten aber eine unbekannte Anzahl

D.h. wir benötigten eine Liste

* Bisher  
  System.Collection.Generic.List<ProtokollEintrag>
* Wir haben aber  
  eine WPF Anwendung
* Ein MVVM Projekt, wo die Datenbindung benötigt wird
* Damit WPF mitbekommt, dass  
  sich der Inhalt einer Eigenschaft   
  geändert hat, steht im Handbuch der Datenbindung.  
  Für Listen ist ObservableCollection<T> zu benutzen

Damit ist für den Protokoll-Auftrag das Thema  
Datentransfer abgeschlossen

* ProtokollEintrag
* ProtokollEinträge

Weiter mit der Klasse, die die Protokolleinträge verwaltet…

* Der ProtokollManager

Problem:

* Wo stellen wir den ProtokollManger bereit?
  + Er wird überall benötigt
  + Er gehört in die Infrastruktur
  + Unsere Infrastruktur ist WIFI.Anwendung.AppKontext
  + Problem  
    In der Assembly WIFI.Anwendung  
    gibt es keinen ProtokollManager,   
    d.h. dort kann der AppKontext nicht geändert werden
  + Lösung  
    Wir müssen den AppKontext,  
    z. B. auf DatenbankAppKontext erweitern

Neues Problem:

* Das bisherige ViewModelAppObjekt  
  beerbt AppObjekt. Dort ist der einfache AppKontext
* Das ViewModelAppObjekt  
  benötigt aber den erweiterten DatenbankAppKontext
* D.h. eine neue Eigenschaft „AppKontext“ wird fällig,  
  die einem das ständige Casten abnimmt

Was benötigt der Protokoll-Manager?

* Methoden zum Hinzufügen von Einträgen

Vorschau

* Das Holen von (externen) Daten kann dauern
* In dieser Zeit ist die Oberfläche „blockiert“
* Windows zeigt unter Umständen „Keine Rückmeldung“ an
* Das wollen wir nicht
* Lösung:  
  Multithreading
* Problem:  
  Ein Thread läuft selbstständig.  
  Kein Thread darf auf die Daten  
  von einem anderen Thread zugreifen
* Deshalb liest man in den Kommentaren  
  vieler .Net Methoden, dass für die   
  Threadsicherheit nicht garantiert wird
* Das kann ein großes Problem werden

Nächster Auftrag

* Damit das Protokollieren vereinfacht wird,  
  soll automatisch registriert werden,  
  wenn ein neues Anwendungsobjekt erstellt wird  
  (NeueInstanz-Eintrag)

Lösung

* Alle unsere Anwendungsobjekte besitzen  
  die Infrastruktur, den AppKontext
* Alle unsere Anwendungsobjekte  
  werden mit Produziere<T>() erzeugt
* D.h. genau dort kann der automatische   
  Protokolleintrag passieren

Problem

* Das Produziere<T> ist in einer Assembly,  
  wo kein Protokoll vorhanden ist
* Das Protokoll ist aber in einer Assembly,  
  die das Produziere<T> bereits verwendet
* UND es ist nicht erlaubt, einen „Zirkelverweis“  
  zu machen

Lösung

* Wir brauchen ein neues Produziere<T>,  
  d.h. Überschreiben in der erweiterten  
  DatenbankAppKontext Infrastruktur

Hinweis:

* Manchmal nutzen Methoden  
  sich selber zur Problemlösung
* Das heißt „Rekursion“
* Damit können oft kurze, elegante Lösung  
  realisiert werden
* Diese sind aber extrem schwer zu Debuggen
* Die Theorie sagt, jeder rekursive Algorithmus  
  kann auch iterativ, d.h. mit Schleifen  
  implementiert werden
* Hat man eine unbewusste Rekursion,  
  stürzt man mit „Stack-Overflow“, einem  
  „Stapelüberlauf“ ab

Nächste Ausbaustufe:

* In der Oberfläche vom Teil 2  
  müssen verschiedene Anwendungsbereiche  
  auswählbar sein, verschiedene „Aufgaben“

Bisher waren wir nur in der Infrastruktur, das Auswählen der Aufgaben ist aber ein Problem vom Teil 2.

Deshalb im WIFI.CS.Teil2 Assembly

* Was ist das?  
    
  eine MVVM Anwendung  
    
  Das Hauptfenster ist eine View.  
  Die WIFI.CS.Teil2.ViewModels.Anwendung ist das ViewModel  
    
  Die Aufgaben gehören jetzt in den „Models“ Bereich

Problem:

* Die neue WIFI.CS.Teil2.Models.Aufgabe  
  hat Eigenschaften, die an die Oberfläche  
  gebunden sind
* Damit die Oberfläche eine Inhaltsänderung  
  mitbekommt, muss PropertyChanged ausgelöst werden
* Die DatenBasis hat das noch nicht

23. Februar 2021

Hinweis zu eigenen Ereignissen:

* Wir wissen
  + Ohne Daten  
      
    System.EventHandler  
    Dieser schreibt System.EventArgs vor
  + Mit Daten   
      
    Eine eigene EventArgs Klasse  
    (oder eine passende Suchen)  
      
    Ein Delegate für die EventArgs
* Zum Mitteilen, dass ein Objekt analysiert wurde,  
  benötigen wir EventArgs, wo ein Objekt transportiert  
  werden kann
* Vielleicht gibt es etwas Passendes,  
  z. B. System.ComponentModel.AddingNewEventArgs

Zu unserm MVVM zurück:

* Im Bereich „Models“ haben  
  wir „Aufgabe“ und „Aufgaben“
* Hier sollen die Anwendungspunkte  
  beschrieben werden
* Die Frage ist, wo?
  + z. B. wie die Sprachen
  + in einer Xml-Datei
  + Eine andere Möglichkeit wäre,  
    die Anwendungspunkte z. B.  
    benutzerbezogen aus einer   
    Datenbank zu lesen
  + …
* Unsere Aufgaben Xml Dateien  
  sollen vor den Benutzern „versteckt“  
  werden, d.h. wir fügen diese, wie  
  die Sprachen in WIFI.Anwendung  
  den Ressourcen hinzu

Wer ist für das physikalische Lesen / Schreiben  
verantwortlich?

* Die Controller
* Wir benötigen einen AufgabenXmlController
* Dieser gehört zum Models-Bereich vom MVVM

Jetzt wollen wir Folgendes:

* Das Holen von Daten ist  
  immer lästig
* Man weiß nie, wie lange das dauert
* Unsere Oberfläche darf  
  aber nicht blockiert sein
* Zum Entwickeln von „ansprechenden“  
  Oberflächen wollen wir  
  den AufgabenXmlController  
  in der Entwicklungsversion  
  künstlich verögern  
    
  System.Threading.Thread.Sleep

Hinweis

* Das „Sleep“ benötigt der Trainer oft
* z. B. bei Web Zugriffen
* Hier muss die eigene Anwendung  
  oft verlangsamt werden, damit  
  sicher eine Rückmeldung kommt

Der Controller (der AufgabenXmlController) ist internal.

Dieser wird von einem ViewModel benötigt

Tipp:

* Nicht alles „in ein ViewModel“
* ABER NICHT FÜR JEDE VIEW UND DETAILVIEW  
  ein eigenes ViewModel
* Thematisch gruppieren

Unser Haupt-ViewModel: WIFI.CS.Teil2.ViewModels.Anwendung

Damit das nicht aus dem Ruder läuft, den Aufgaben-Teil  
in einem ungeordneten AufgabenManager verstecken

Ein Objekt vom AufgabenManager wird im Haupt-ViewModel veröffentlicht

Damit die Anwendungen.Aufgaben.Liste „visualisiert“ wird, diese Eigenschaft in einer View binden

Tipp

* Nicht alles in einer Xaml Datei
* Das ist so und so schon unübersichtlich genug
* Deshalb: Ein eigenes AufgabenView.xaml

Problem

* Das Holen der Daten, bei uns künstlich  
  Anwendung.Aufgaben.Liste,  
  dauert
* In dieser Zeit ist die Oberfläche blockiert
* Für die View benötigen wir eine Eigenschaft
* Damit nicht blockiert wird, muss  
  das Holen der Daten in einen eigenen Thread

Was ist ein Thread?

* Ein „Unterprozess“ vom Hauptprozess, der Anwendung
* Ein Thread wird vom Betriebssytem  
  auf einem Kern des Prozessors ausgeführt
* Ein Thread läuft selbstständig
* Wird eine Anwendung auf mehrere Threads  
  aufgeteilt, spricht man von „Multithreading“
* Das ist eine der schwierigsten Aufgaben überhaupt
* KEIN THREAD DARF IN DIE DATEN  
  VON EINEM ANDEREN THREAD GREIFEN!
* Weil das so schwierig ist, hat Microsoft  
  ein neues Programmiermuster zum Arbeiten mit Threads  
    
  [„Aufgabenorientiere asynchrone Programmiermuster“](https://docs.microsoft.com/de-de/dotnet/standard/async)  
  „Task oriented Programm Pattern“, kurz TAP  
    
  Implementiert in …  
    
  System.Threading.Tasks

Zum aktuellen Problem

* Weil eine Eigenschaft nicht in einem Thread laufen kann,  
  muss eine eigene Methode zum Holen der Aufgaben  
  asynchron in einem eigenen Thread erstellt werden

Lösung

* AufgabenManager.InitialisiereAufgabenAsync

Zurück zur Einführung:

* Man kann gar nicht zu viel protokollieren
* Wir haben jede Methode, z. B. in Algorithmus-Klasse  
  mit „Startet…“ und „Beendet“ versehen

Diese Tugend wollen wir wieder aufnehmen

* Vor allem wir das benötigt,  
  damit die Anwendung anzeigen kann,  
  dass sie beschäftigt ist
* Jetzt im Teil 2 soll das „Startet…“  
  und „Beendet“ aber etwas vereinfacht werden

Lösung:

* Soweit möglich, in einer der Basisklassen,  
  nicht in jedem Anwendungsobjekt neu
* D.h. in ViewModelAppObjekt
* Eine IstBeschäftigt-Eigenschaft

Problem

* An vielen Stellen kann das IstBeschäftigt  
  aktiviert und deaktiviert werden
* D.h. wir müssen mitzählen, wie oft  
  IstBeschäftigt gesetzt wurde und  
  wie oft IstBeschäftigt deaktiviert wurde