Daueraufträge:

1. Die C# Schlüsselwörter auswendig lernen  
   <https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/>
   1. Modifizierer
   2. Datentypen
2. Die C# Operatoren auswendig lernen  
   <https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/operators/>

Meisinger Klemens  
WIFI OÖ GmbH

3882 C# Teil 1 – Komponenten

Teil von 3881 -> C# Komplettausbildung

* Bisher: 3880 Einführung in die objektorientierte Programmierung (C# und .Net)
* 3883 C# Teil 2 – Schwerpunkt WPF und Datenbank
* 3884 C# Projekt, 2er Gruppen, und Prüfung

Zur Prüfung:

* Hier kann man nicht 14 Tage vorher zum Lernen anfangen

Tipp:

* „Kurstagebuch“ (Wissenstransferbuch) anlegen
  + Hier immer die Schlagwörter eines Abends niederschreiben
  + Fragen notieren
  + In der nächsten Sitzung die Fragen stellen
* (Einladung) Hausübungen machen
  + Prüfung hat einen Theorieteil (offene Verständnisfragen, alle Unterlagen erlaubt)
  + Prüfung hat einen Praxisanteil
    - 4 Beispiele
    - Eines kann gestrichen werden
    - Hausübungen sind mit den Prüfungsbeispielen vergleichbar

Termine:

Do 29.11.18

[Di 04.12.18](#Zwei)

[Do 06.12.18](#Drei)

[Di 11.12.18](#Vier)

[Do 13.12.18](#Fünf)

[Di 18.12.18](#Dez18)

[Do 20.12.18](#Dez20)

[Do 17.01.19](#Jan17)

[Di 22.01.19](#Jan22)

[Do 24.01.19](#Jan24)

[Di 29.01.19](#Jan29)

[Do 31.01.19](#Jan31)

Teilnahmebestätigung nur bei 75 % Anwesenheit

Zeitplan:

17:30 – 18:45

19:00 – 20:15

20:30 – 21:45

Teilnehmerinnen und Teilnehmer

BAUER Angelika

~~BITZINGER Sarah~~

DUSANIC Marco

FELBER Jörg

GRABNER Dominik

KRONAWETTLEITNER Robert

LEITNER Niklas Matthias

NIEDERMAYR Wolfgang

PIRKLBAUER Hermann

SCHLAGER Jürgen

SCHWARZ Dietmar

SPECKHARD Christoph

STIEGLBAUER Selina

ZENKOVIC Damir

Zum Inhalt des Inhalt Trainings:

* Microsoft Handbuch zur C# Programmierung  
  <https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/>

Microsoft Nachschlagewerk

* Früher: MSDN (msdn.microsoft.com/library)
* Neu: docs.microsoft.com

Beim Tippen mit C# wird .Net entwickelt

* Was ist .Net?  
    
  Eine Klassenbibliothek (ein Framework)

Was ist ein Klasse?

* Vergleichbar mit einem Hausplan.  
  Den Hausplan „genehmigen“ („statisch“, singleton im Fachbegriff)

Zum Benutzen einer Klasse?

* Ein Objekt instanzieren
  + Zur Beschreibung: Eigenschaften
  + Zum Arbeiten: Methoden
  + Für Mitteilungen: Ereignisse

Wie läuft das Training?

* Möglich:  
  Jeden Abend ein neues Project für ein Spezialthema  
  so sind die Bücher aufgebaut
* Hier lernt man nix.  
  Hier kann man einzelne Fragmente.  
  Diese aber nicht im Zusammenhang benutzen
* Vom Trainer bevorzugt:  
    
  EIN Projekt für eine Gesamtaufgabenstellung,  
  das die Lernziele enthält

Zum Projekt:

* Dieses Training ist für Personen, die bereits programmiert haben
* Programmieren ist „Nachdenken“
* Nicht verwechseln mit Tippen
* Beim Nachdenken muss man einen
  + Algorithmus finden
  + Nur drei Bausteine
    - Sequenz
    - Verzweigungen
      * Binär (if)
      * Fall (switch)
    - Schleifen
      * Zählschleife (for oder foreach)
      * Abweiseschleife (while)
      * Durchlaufeschleife (do)

Auftrag:

* Eine grafische Anwendung (mehrsprachig) ist zu erstellen, womit sich eine Person, die bereits programmiert, darüber informieren kann, wie mit C# getippt wird.

Fertigstellungstermin:

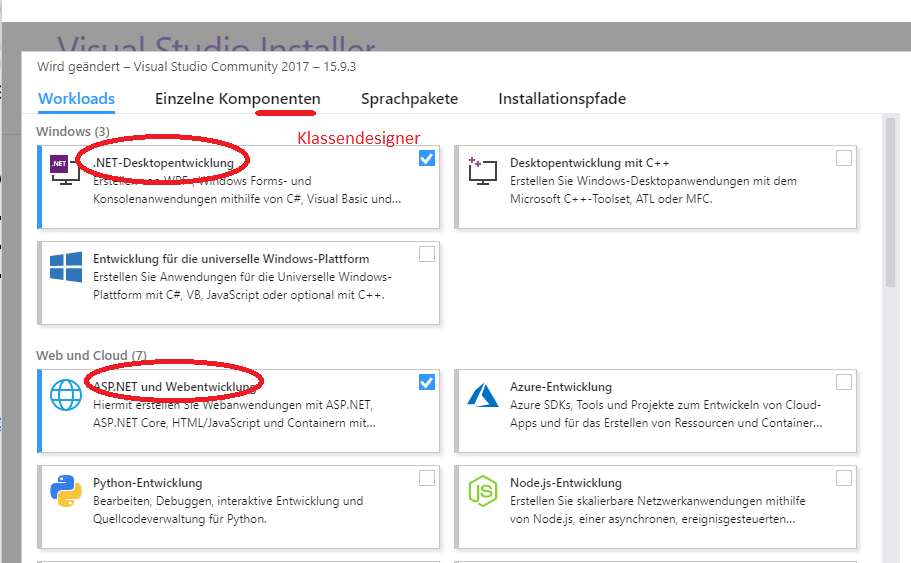
Do 31.01.19

Entwicklungsumgebung:

* Microsoft Visual Studio 2017 (Community)
* Kostenlos, Microsoft Konto erforderlich
* Aus dem Internet:  
    
  Visual Studio Installer

<https://visualstudio.microsoft.com/de/downloads/>

Für die Reihe 3881 ist notwendig…



Zum Auftrag:

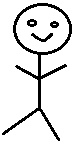
* Eine grafische Anwendung (mehrsprachig) ist zu erstellen, womit sich eine Person, die bereits programmiert, darüber informieren kann, wie mit C# getippt wird.

Wie läuft so ein Projekt ab?

Am Anfang:

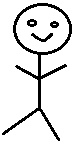
* Die Analyse
* Heute „objektorientiert“  
  (nicht wie früher sequentiell, funktionale Top-Down Zerlegung)
* Womit wird die Analyse beschrieben?
  + Mit der „Unified Modeling Language“ (UML)
  + Quelle für die UML bei der International Business Machines Corporation (IBM)  
      
    <https://www.ibm.com/developerworks/rational/library/769.html?mhq=uml%20basic&mhsrc=ibmsearch_a>
* Ergebnis der Analyse beschrieben mit der UML  
    
  Klassendiagramm

Zur UML:

1. The Actors Diagramm (Benutzerdiagramm)

Eine Person, die  
bereits programmiert hat

1. The Uses Cases Diagram (Anwendungsfälle, je Actor)



Allgemeine Information

Aufbau einer C# Datei

Zählschleife  
(for)

Fallent-scheidung  
(switch)

Binärent-scheidung (if)

Abweise-schleife (while)

Durchlaufe-  
schleife (do)

1. The Scenarios (Szenarien)

|  |  |
| --- | --- |
| Anwendungsfall | Szenario |
| Allgemeine Information  (bis Durchlaufeschleife) | Inhalt aus einer Textdatei anzeigen. Der Benutzer möchte unter Umständen…   * Kopieren * Speichern * Drucken * Zurück- bzw. Vorwärtsgehen * Sprache wechseln |
| Bei allen anderen Fällen identisch | |

1. The Sequence Diagram (der zeitliche Ablauf der Benutzung)

Start der Anwendung

Die Sprache von der alten Sitzung wiederherstellen

Die Fensterposition und –zustand von früher wiederherstellen

Anwender begrüßen (Begrüßung)

Abhängig von einer Auswahl

Information anzeigen

Passende Textdatei lesen

Sprache wechseln

Kopieren

Speichern unter

Drucken

Zurück- oder vorwärts Blättern (Verlauf)

Beenden

Den Benutzer fragen, ob sicher

Falls ja

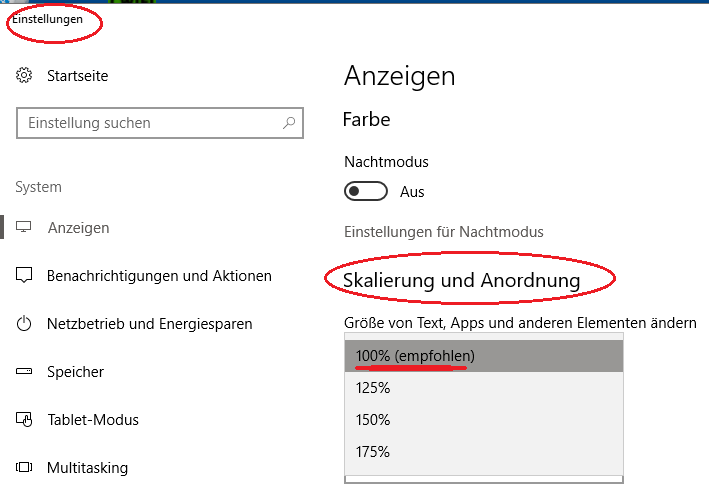
Die aktuelle Sprache merken

Die aktuelle Fensterposition   
 und –zustand merken

* Die UML kennt viele weitere Darstellungen
* Für unseren Fall reicht das aber

--------

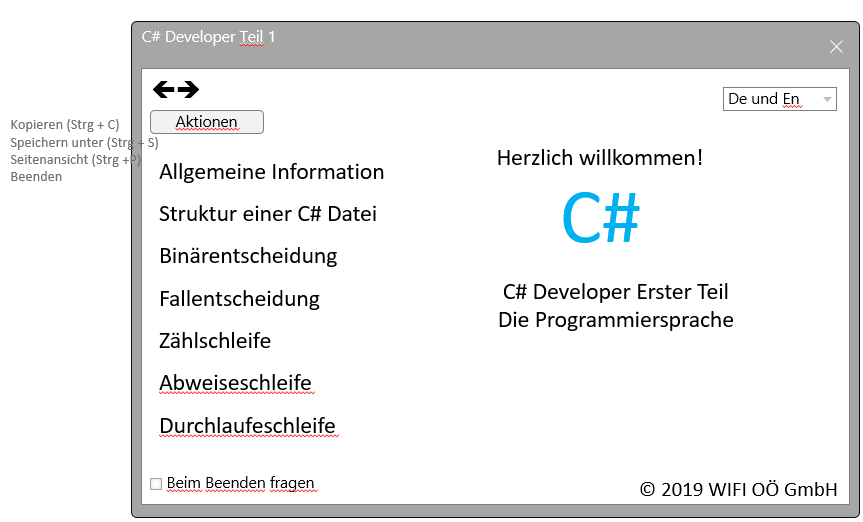
Hier kann dann ein Prototyp der Anwendung gezeichnet werden

* Dazu muss beim Auftraggeber nachgefragt werden
* Welche grafische Oberfläche?
  + Was für Möglichkeiten bestehen?
    - **Webanwendung** (ASPX, Active Server Pages)  
      Vorteil: Kann überall in einem Internet Browser laufen  
      Nachteil: Provider  
      Nein, hier nicht
    - **Universelle Windows Plattform**  
      Vorteil: Läuft auf allen Geräten, wo Windows als Betriebssystem benutzt wird. Diese Anwendung kann in den App-Store gestellt werden  
      Nachteil: Teilzugriff auf das .Net Framework
    - **Windows Forms**Nachteil: Konzipiert für Windows XP. Die Oberfläche ist heute nicht mehr „scharf“. Nur bei einer Skalierung von 100%  
        
      Vorteil: Einfacher beim Entwickeln  
        
      Hauptgrund für den Trainer:  
      In vielen Firmen gibt’s eine Vielzahl an Windows Forms Anwendung. Man sollte es zumindest gesehen haben.
    - **Windows Presentation Foundation (WPF)**Vorteil: Nutzt die Grafikkarte des Rechners direkt (DirektX). Bei allen Skalierungen scharf. Es kann das ganze .Net benutzt werden

Nachteil: Relativ komplex beim Entwickeln der Oberfläche

Zum Prototyp Entwickeln:

* PowerPoint (ab 2013. Teil vom Office Professional Plus)



Ergebnis der Analyse:

* Klassendiagramm

Wie finden wir diese?

* Grundsätzlich
  + Die Hauptsubstantive sind Klassen
  + Für jede Aufgabe, eine Klasse. Klassen nicht überladen
  + In unserem Fall:
    - Anwendung
    - Sprache
    - Fenster
    - Begrüßung
    - Information
    - Verlauf

Klassen werden in

* Komponenten

eingeteilt.

Grundprinzip:

* Wiederverwendbaren Code, Klassen, erzeugen

Eine Komponente ist in .Net eine …

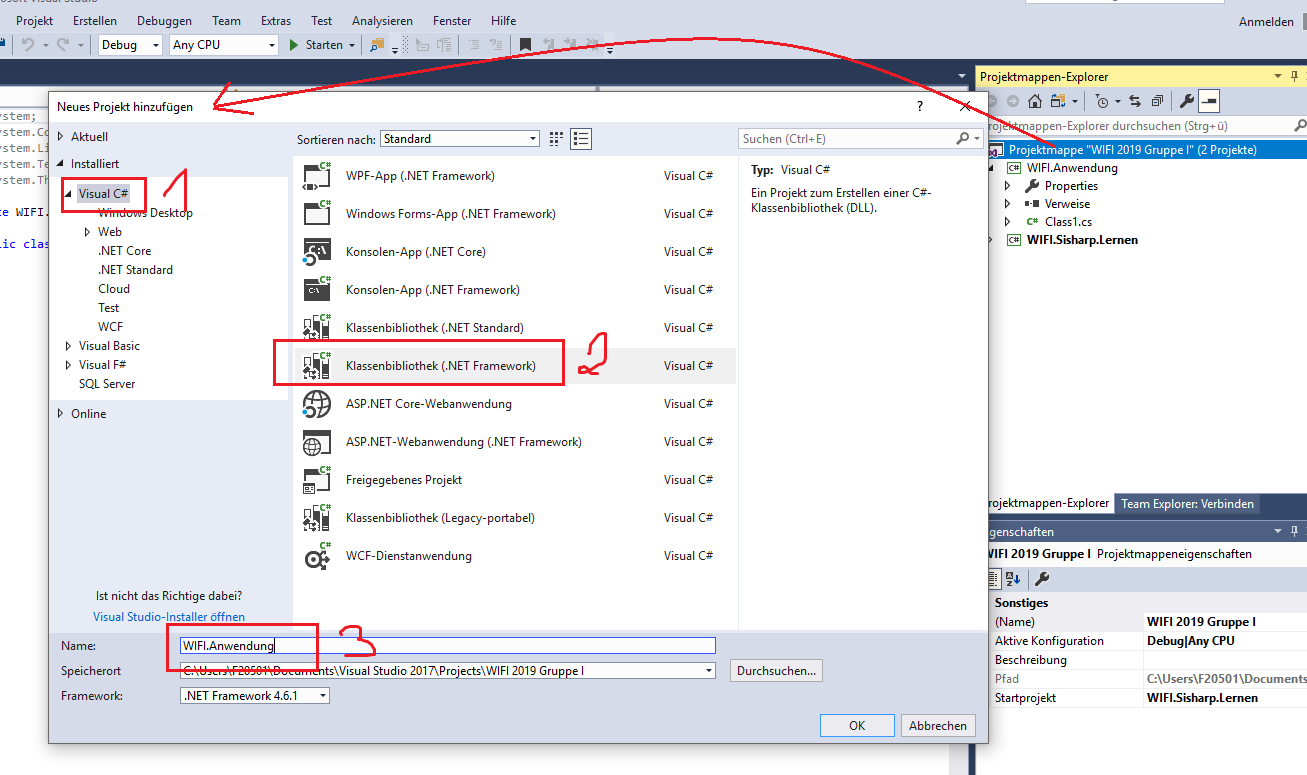
* Assembly  
  (Ein Projekt, wenn das Ganze gerade erstellt wird)

Wichtig:

* Jede Firma muss zu einem eigenen Framework kommen
* Damit bei einem neuen Projekt nicht wieder von vorne begonnen wird
* Wir teilen die Klassen auf zwei Komponenten (Assemblies)
  + Eine Assembly für Klassen, die auch im Teil 2 benötigt werden
  + Eine Assembly für das, was nur den Teil 1 betrifft
* Derselbe Klassename darf im gesamten .Net nur einmal vorkommen, muss eindeutig sein
* Deshalb gibt’s Konventionen zum Benennen von Assemblies

Zu unseren Assemblies

1. Für das „Firmenframework“  
     
   WIFI.Anwendung  
     
   In diesem Projekt, in der Assembly sollen Klassen sein, die in allen Anwendungen benötigt werden.  
   Eine Klassenbibliothek



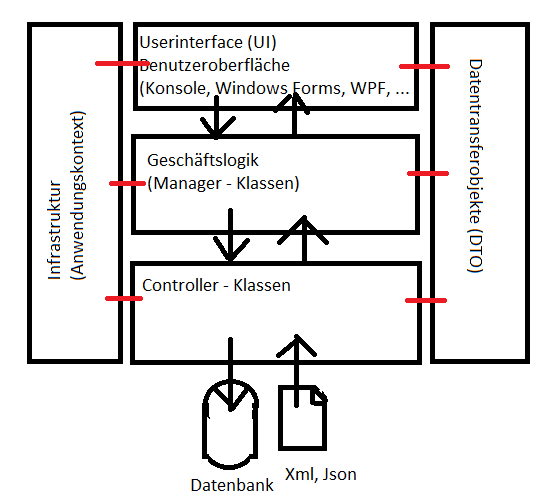
1. Für den Teil 1  
     
   WIFI.Sisharp.Teil1  
     
   Dieses Projekt soll „ausführbar sein“. Eine Windows Anwendung.  
   Mit der Windows Forms Plattform.

Grundsätzlich gilt:

* Mehrschichten Architektur

<https://de.wikipedia.org/wiki/Schichtenarchitektur>

Schema



Anwendungen (die meisten) verarbeiten Daten. Diese befinden sich in Datenbanken (Teil 2) oder in „selbstbeschreibenden“ Textdateien (Xml, Json)

<https://de.wikipedia.org/wiki/JavaScript_Object_Notation>

Die Klassen, die direkt die Daten schreiben oder lesen, sind die Datenhaltung, die Controller. Im Training wird notwendig ein XmlController zum Speichern und Lesen der Fensterpositionen.

Die Klassen, die die Daten auf Plausibilität prüfen bzw. berechnen, usw… sind die Klassen der Geschäftslogik. Bei uns z. B. „FensterManager“.

Die Geschäftslogik wird visualisiert in der Benutzeroberfläche (UI).

Grundprinzip der objektorientierten Programmierung:

* Die Datenkapselung

Im Idealfall ist jede Schicht eine eigene Assembly.

Schwierigkeit:

* Es muss ein Objekt geben, das überall bekannt ist
* Der Anwendungskontext, die Infrastruktur, notwendig

Die Klassen, in den „nur“ Information transportiert wird (über die Ebenen)

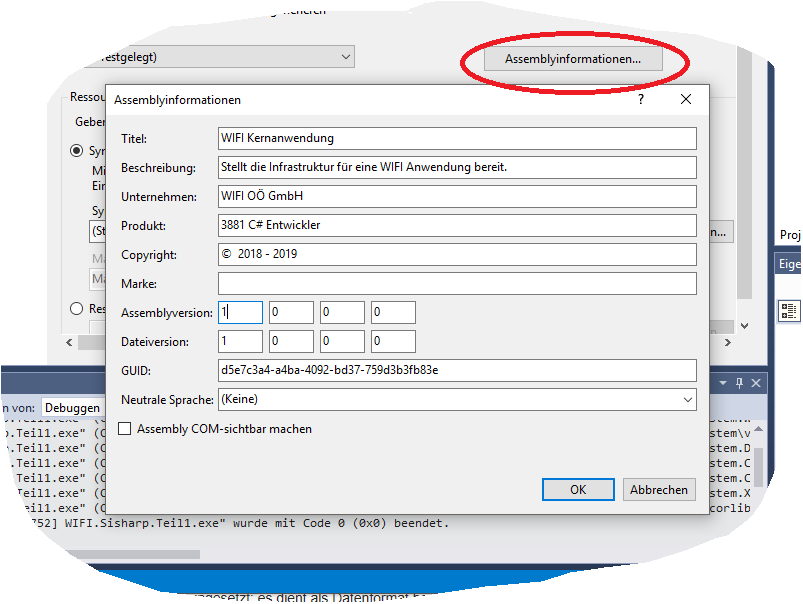
* Datentransferobjekte (DTO)

Grundsatzfrage zur Sprache der eigenen Klassen (Objekte):

* Welcher Amerikaner spricht freiwillig deutsch?
* Warum sind wir so deppert und reden „Englisch“?
  + Es gibt keinen Grund
  + Außer  
      
    Eine Firma, die weltweit agiert, z. B. Fronius,   
    wo die Firmensprache Englisch ist
  + Die Firmensprache vom WIFI ist deutsch
  + Deshalb wird das Klassenmodell ebenfalls deutsch
  + Wichtig:  
      
    Auf KEINEN FALL mischen!!!

Zum Umsetzen:

* Das Erste in einer .Net Assembly, in einem Projekt
* Die Eigenschaften einstellen



* Besonders die „Assemblyinformation“

Zur Version:

4 Teile

1. Hauptnummer  
     
   Diese ändert sich, wenn eine andere Technologie eingesetzt wird, z. B. von Windows Forms auf WPF
2. Nebennummer  
     
   Diese ändert sich, wenn neue Klassen ergänzt werden.
3. Nummer  
     
   Diese ändert sich, wenn Eigenschaften, Methoden, … verändert wurden.
4. Revision  
     
   Ändert sich, wenn Fehler ausgebessert wurden.

Oberster Grundsatz beim Tippen:

* Es darf nichts, wirklich nichts, ohne Kommentar geben

C# unterstützt alle Kommentarversionen der „C-Welt“

Original

/\*

Blockkommentar

\*/

Mit der objektorientierten Version von C/C++

// Wird am Zeilenende

// beendet.

Mit C# und .Net wurde ein neuer Kommentar eingeführt:

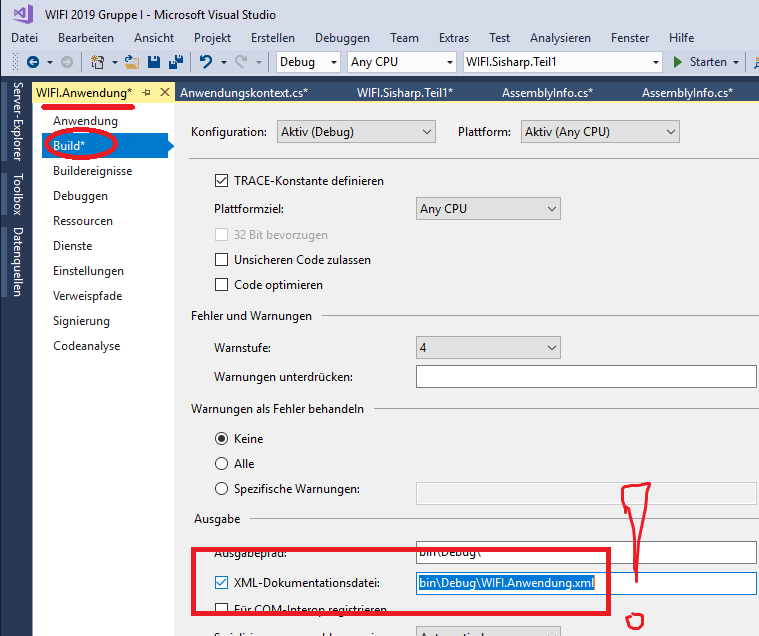
<https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/xmldoc/xml-documentation-comments>

Dieser wird mit drei Schrägstrichen über einer Deklaration gesetzt.

WICHTIG:

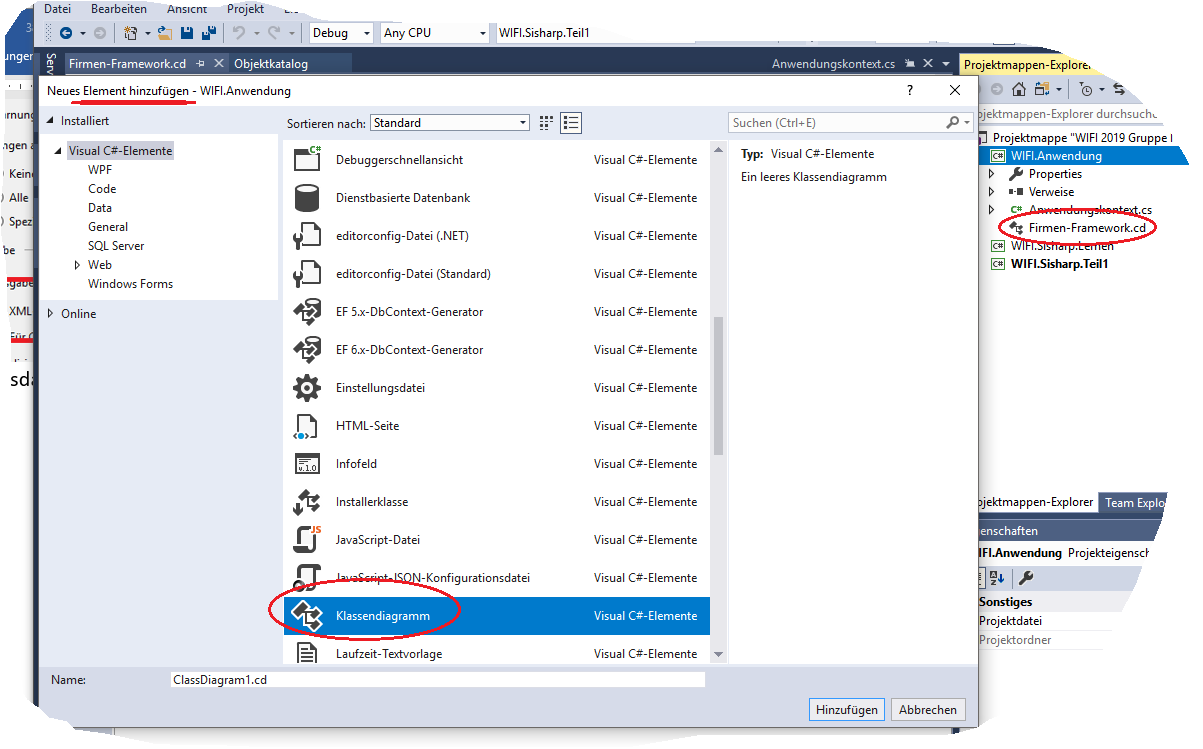
* Der Kommentar ist IMMER DAS ERSTE, was getippt wird!!!!
* Im Kommentar steht, was passiert, und nicht wie etwas passiert

Hinweis:

* Damit die Kommentare für andere „mitkompiliert“ werden,   
  in den Eigenschaften…  
    
  … „XML-Dokumentationsdatei“ aktivieren

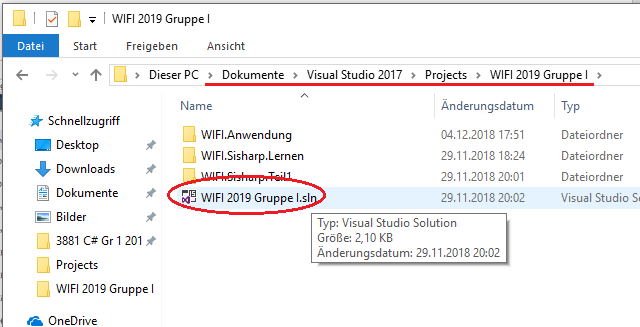
4.12.2018

Visual Studio kann das Ergebnis der Analyse …

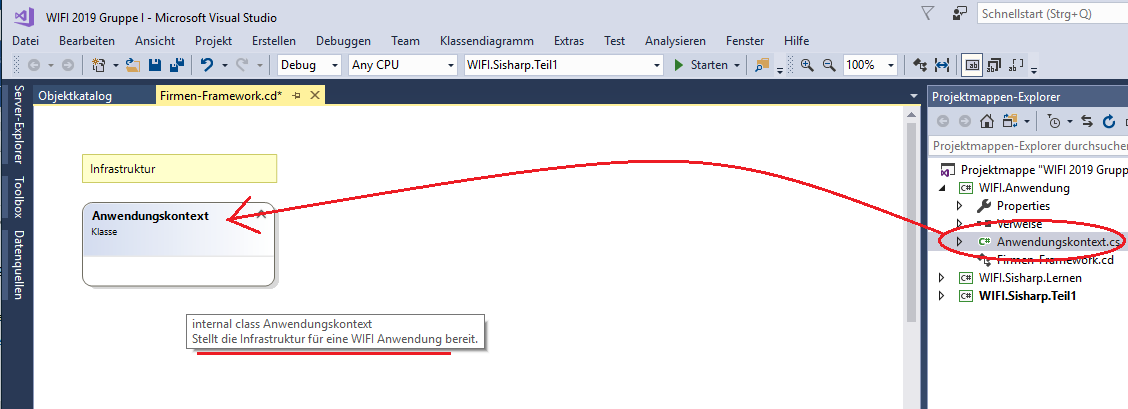


… als Klassendiagramm darstellen. Muss im Installer unter Einzelkomponenten extra eingeschaltet werden.

Erinnerung:

* Sie dürfen den Projektordner NICHT ändern, also umbenennen oder sonst etwas
* Wir arbeiten lokal über den Standardordner  
  
* Sicherung ist auf Q:
* Falls diese benötigt wird:
  + Original auf C: löschen   
    (damit falsche Dateien, … sicher entfernt werden)
  + Von Q: den Ordner aus dem Zip  
    lokal einfügen

Zum Klassendiagramm



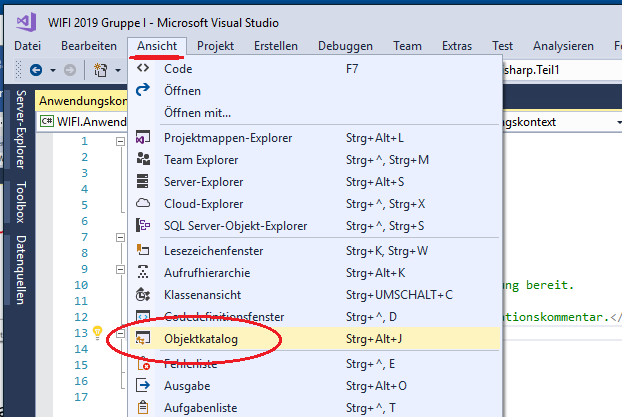
Hier können Kommentare über das Kontextmenü und Dateien aus der Projektmappe positioniert werden.

* Im Tool-Tipp kann der Xml Dokumentationskommentar gelesen werden.

Zur .Net Entwicklung

* Jedes neue Element ist eine Erweiterung von  
  System.Object
* System.Object ist die erste Klasse von Microsoft. Deshalb der Standard Datentyp

Alle vorhandenen Klassen können …



… im Objektkatalog durchforstet werden.

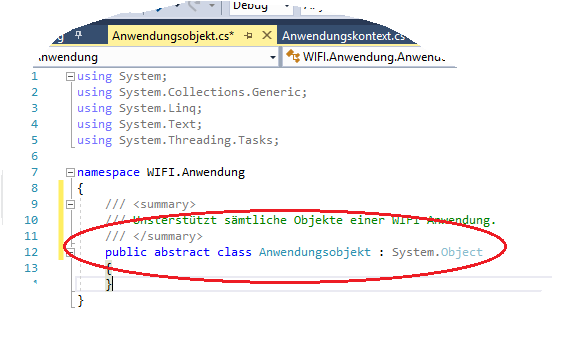
Erste Assembly  
  
mscorlib

Erste Ebene Namespace {}  
  
System

Erste Klasse  
  
Object

Tipp:

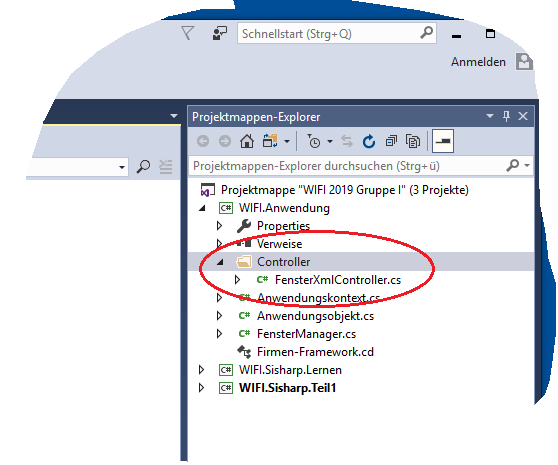
Zwischen System.Object und …



… den eigenen Klassen eine Zwischenebene einfügen, für Eigenschaften und Methoden, die überall benötigt werden.

Tipp:

* Frühzeitig beginnen, die eigenen Klassen zu gliedern
  + „Namespaces“
  + Visual Studio und C# erstellt automatisch  
    Namespaces, wenn der Projektordner  
    in Unterordner aufgeteilt wird



Hauptproblem in der Entwicklung:

* Eine unbekannte Anzahl an Daten

Es können nicht beliebig viele Variablen deklariert werden.

Es muss EINE Variable für mehrere Werte benutzt werden…

* Datenfeld, Array oder Liste

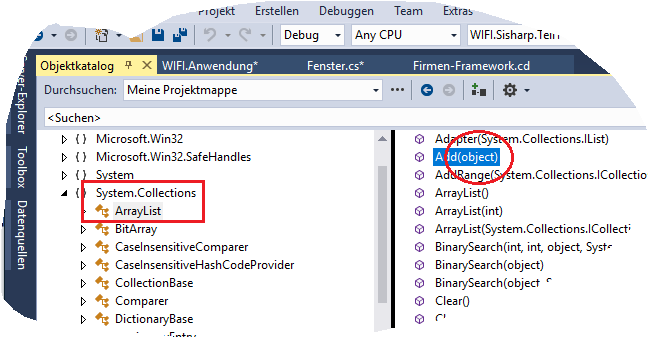
C# hat nur statische Arrays, also Datenfelder, die nach der Deklaration im Umfang nicht mehr verändert werden können.

Weil die Anzahl der Fenster nicht bekannt ist, benötigen wir…

* dynamische Datenfelder

Weil C# das nicht kann, Lösung…

* .Net



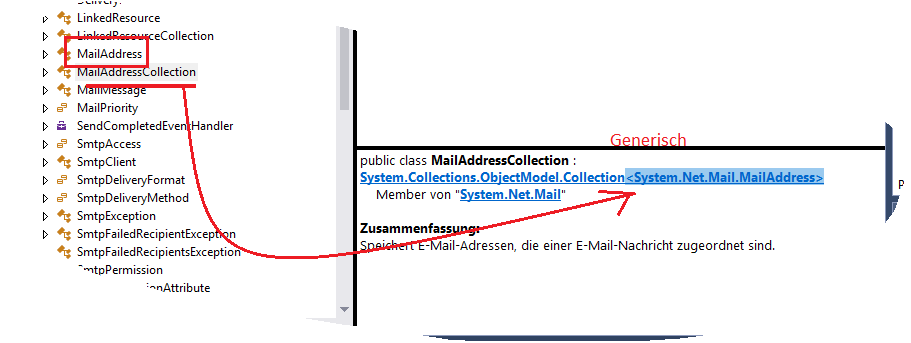
Hier befindet sich z. B. in System.Collections die ArrayList.

Nachteil von dieser dynamischen Liste:

* Sie benutzt den Standardtyp System.Object
* Deshalb kann alles hinzugefügt werden
* Beim Abrufen muss aber der Typ bestätigt werden
* Diese „Typ-Bestätigung“ heißt „Casten“
* Das ist nicht lustig

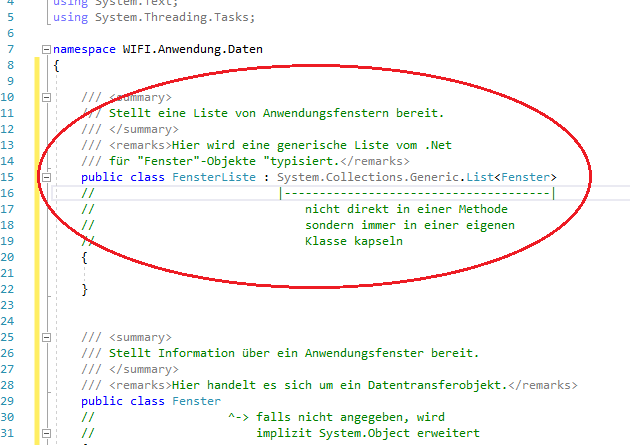
Wir wollen „stark typisierte“ Listen

Was macht Microsoft in .Net:



* Es wird eine Basisklasse, die ohne Typ implementiert wurde, „typisiert“
* Diese wichtige Technik heißt
* Generische Programmierung

<https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/generics/>



Zu den Datentypen:

* Jede eigene Klasse kann als Typ benutzt werden

Grundsätzlich kennt .Net

* Zwei Datentyp-Familien

Das sind

1. Verweistypen  
   Hier befindet nur eine Speicheradresse der Daten.  
   Implementiert als „Klassen“.  
   Keine Speicheradresse: null
2. Werttypen  
   Hier befindet sich die Information direkt in der Variable.  
   Implementiert als „Strukturen“.  
   Werttypen können nicht „null“ sein. Die Standardinitialisierung für Werttypen ist 0 (die Zahl Null).

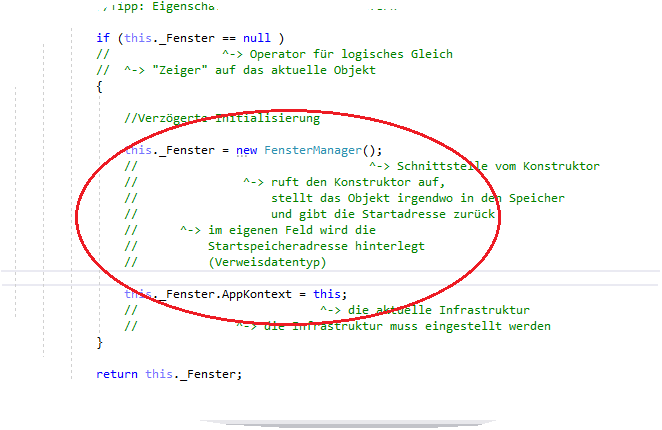
Mit Klassen macht man nie etwas falsch. Strukturen passen nicht immer.  
D.h. heute grundsätzlich Klassen.

WICHTIG! WICHTIG…

* Wir benötigten von einer Klasse ein Objekt
* C# benutzt dazu das „new“ Schlüsselwort
* Das Initialisieren ist relativ langsam
* NICHT ALLE MÖGLICHEN OBJEKTE BEIM STARTEN INITIALISIERN  
  (Keine Adobe Anwendungen)
* Das Initialisieren verzögern, „Just-In-Time“ und…
* Cachen

<https://docs.microsoft.com/de-de/dotnet/framework/performance/>

Außerdem:

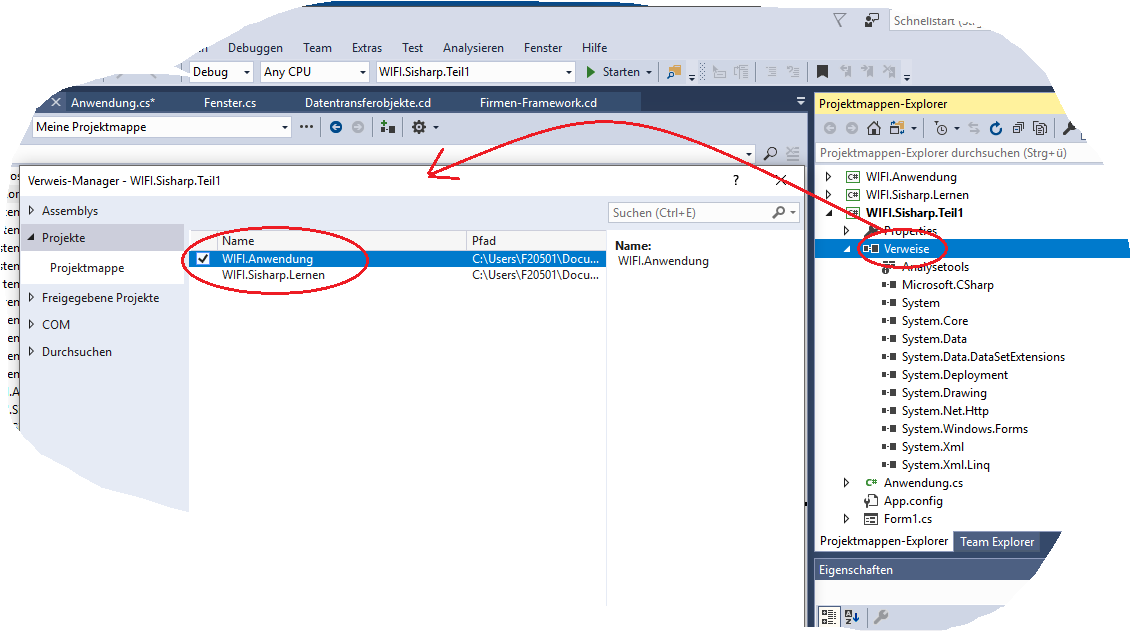


* Wir haben keine Lust, bei jedem Anwendungsobjekt die Eigenschaft „AppKontext“ einzustellen
* Wir wollen eine Methode zum Initialisieren, zum Erzeugen eines Anwendungsobjekts (eine Objekt-Fabrik)

Zum Haupteinstieg in eine .Net Anwendung:

* Main() Methode
* Am Anfang gibt’s noch keine Objekte
* Deshalb muss Main() direkt einer Klasse gehören
* Mitglieder, die einer Klasse (nur EINMAL) gehören,  
  sind „statisch“

Wie kann eine Assembly eine andere nutzen?



6.12.2018

Vor ein paar Jahren hat’s einen Philosophie – Wechsel bei Microsoft gegeben

* Früher  
    
  „Run By Default“  
  Auf Deutsch: Und wenn jemand überhaupt nix weiß, muss das Einlegen eine Microsoft Software CD reichen, damit die Anwendung läuft.  
    
  Das führte dazu, dass Microsoft Anwendungen „als nicht sicher“ gehandelt wurden.  
    
  Deshalb alle Deklarationen „Public“
* Heute:  
    
  „Secure By Default“  
    
  Auf Deutsch: Microsoft Anwendungen müssen sicher sein.  
  Das führt dazu, dass heute oft ohne vorherige Sicherheitseinstellungen eine Microsoft Anwendung nicht sofort läuft. -> Man muss was wissen.  
    
  Alle Deklarationen „Private“

Tipp:

* Grundsätzlich  
    
  „Voll qualifizieren“, d.h. nicht „Anwendungskontext“ sondern „WIFI.Anwendung.Anwendungskontext“
* Warum funktioniert „Anwendungskontext“ überhaupt?  
    
  Wegen „zu faul zum Tippen“, der „using“ Direktive
* Konventionen zum „var“  
    
  <https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>

Faustregel zu Eigenschaften und Methoden

* Oft ist nicht selbstverständlich, ob eine Eigenschaft oder Methode
* Es gilt:  
    
  Wird etwas „beschrieben“ und steht „sofort“ bereit…  
    
  Eine Eigenschaft, Substantiv, Pascal-Notation  
    
  Wird etwas „berechnet“ …  
    
  Eine Methode, Verb, Pascal-Notation

Zu den Sprachen:

* Wir müssen Deutsch und Englisch als Oberflächensprache unterstützen.
* Später unter Umständen noch mehr
* Das muss irgendwo hinterlegt werden

Es bieten sich „selbstbeschreibende Textdateien“ an…

z. B. Extensible Markup Language (Xml) oder JavaScript Object Notation (Json)

Zum Xml, weil das Visual Studio selber alles als Xml beschreibt

<wurzelElement>

<element>

<element attribut="Einstellung"/>

</element>

</wurzelElement>

Konvention:   
Die Namen der Xml-Elemente und Attribute sind klein geschrieben.

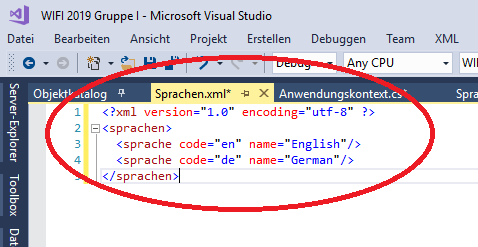
Zum Entwickeln von mehrsprachigen Anwendungen:

* Als „Standardsprache“ ist die zu wählen, die benutzt werden soll, wenn’s keine Übersetzung
* Heute Englisch

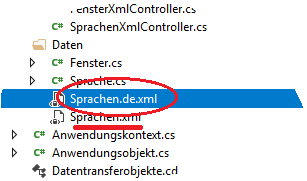
Zu mehrsprachigen Anwendungen:

* Im .Net „ganz unten“ verwurzelt
* Die erste Assembly  
    
  mscorlib  
  System.Globalization.CultureInfo

Der Inhalt einer eigenen Xml Datei ist frei wählbar…



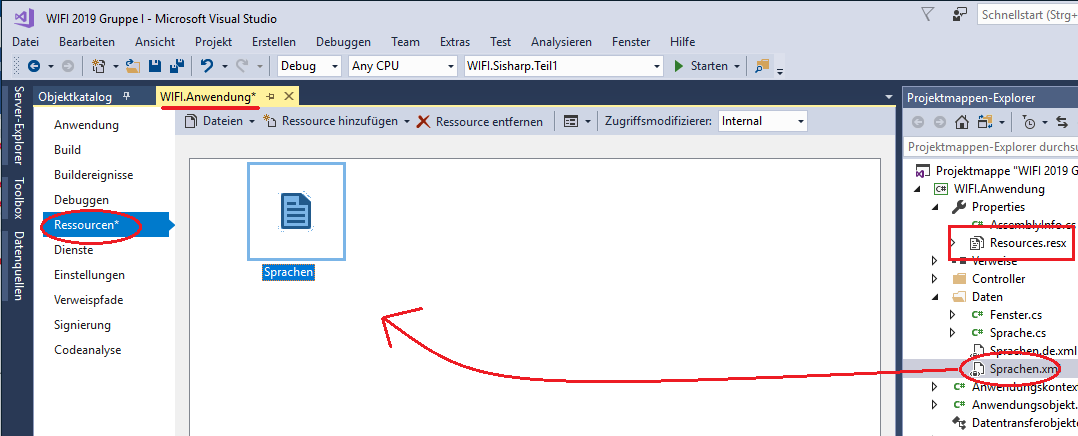
Wie lokalisiert, übersetzt, Microsoft?



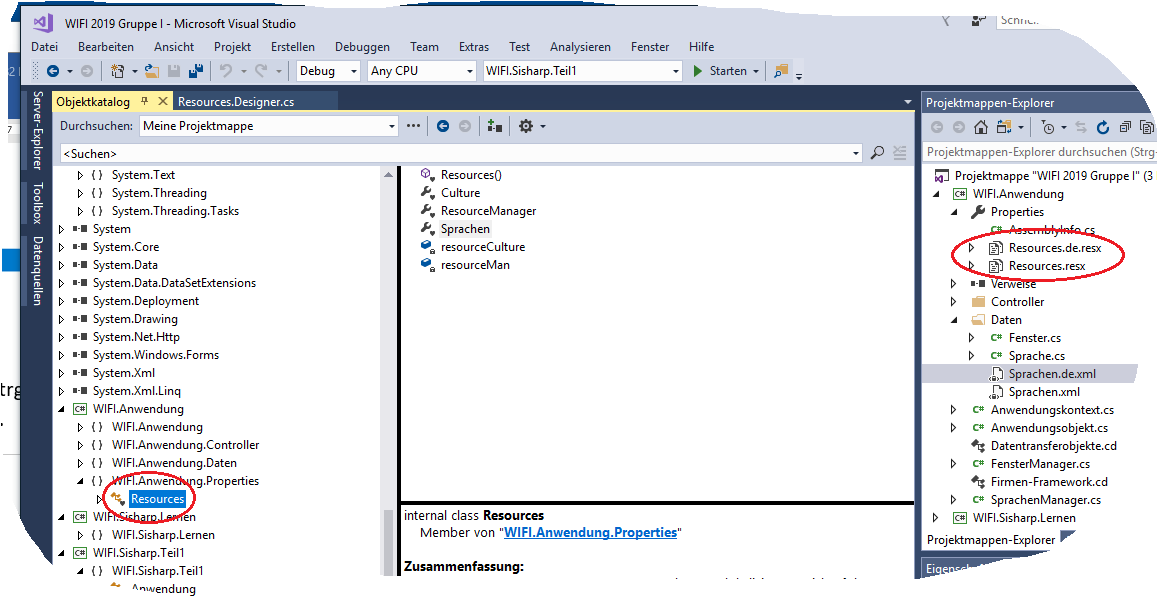
Die originale Datei wird kopiert (Strg + C und Strg + V) und vor der Erweiterung der CultureInfo Code geschrieben.

Wo kann Zusatzinformation für eine Anwendung, außer direkt als Dateien im Anwendungsverzeichnis, hinterlegt werden?

* In den Anwendungsressourcen (Projekteigenschaften)

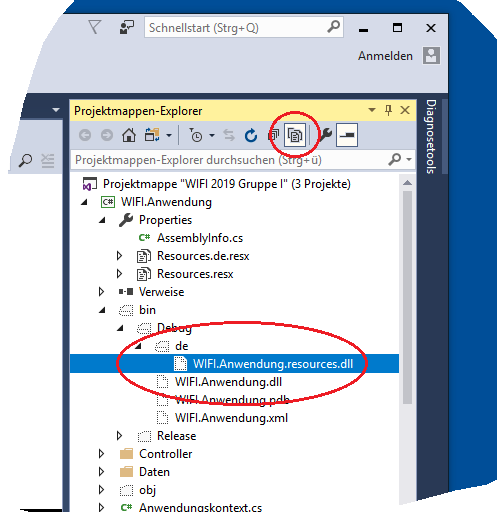


Die Ressourcen werden vom Studio über eine automatisch erstellte Klasse …



… im Properties Namespace bereitgestellt.

Beim Kompilieren …



… werden lokalisierte Ressourcen in „Satelliten Assemblies“ hinterlegt.

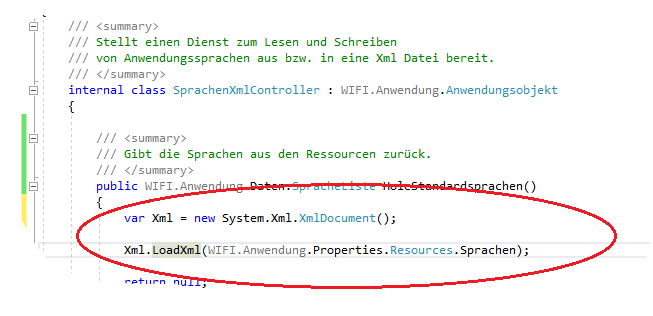
Mitglieder von Klassen:

* Zur Beschreibung  
    
  Eigenschaften, z. B. Sprache.Code, Sprache.Name, …  
    
  Substantiv, keine runde Klammern (keine Schnittstelle)
* Zum Arbeiten  
    
  Methoden, z. B. SprachenXmlController.HoleStandardsprachen()  
    
  Verb, runde Klammern für die Schnittstelle der Methode

Noch nicht erwähnt: Ereignisse für Mitteilungen

C# hat nix zum Verarbeiten von Xml-Dateien

* Wir hoffen auf .Net



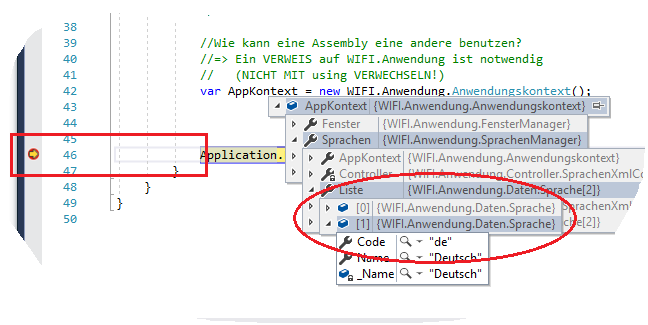
Die Abbildung zeigt einen Einstieg in System.Xml. Außerdem wie über „Properties.Resourcen“ auf die definierten Ressourcen zugegriffen wird, weil vom Studio automatisch (statische) Eigenschaften implementiert werden.

Zugriff auf EIN Element eines Arrays…

<https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/indexers/>

* Eckige Klammern, z. b. SpracheListe[0] für das erste Element
* DAS GESAMTE .NET IST NULLBASIEREND, d.h. das erste Element wird mit Index 0 angesprochen

Nachdem dem SprachenManager eine Eigenschaft für die Liste der Sprachen beigebracht wurde, kann …



… beim Debuggen der Inhalt vom AppKontext durchforstet werden.

Debuggen (Gezielte Fehlersuche in einer Anwendung):

* Normal laufen lassen: F5
* Schritt für Schritt: F11
* Methoden als ein Schritt: F10
* Haltepunkte umschalten mit F9
* Alles im Menü Debuggen

Die Abbildung zeigt, die Anwendung benutzt bei den Sprachen „Deutsch“. Warum? Weil das Betriebssystem Deutsch ist

* Der Benutzer muss aber die Sprache auswählen können

Wo ist die Sprache für eine Anwendung hinterlegt?

* Im aktuellen Thread

11.12.2018

Frage:

* Warum hat unsere Infrastruktur eine Anwendungskontext – Klasse und eine Anwendungsobjekt – Klasse?

In .Net kann eine Klasse eine andere Klasse erweitern (beerben). Dabei gibt’s aber nur eine Einfachvererbung. Wird keine Basisklasse ausdrücklich angegeben, wird System.Object erweitert.

Alle Objekte benötigen bei uns die Eigenschaft AppKontext. Diese AppKontext Eigenschaft ruft das Anwendungskontext-Objekt ab. Wenn jetzt alle die Objekte AppKontext benötigen, muss diese überall implementiert werden. Keine Lust dazu.

* Deshalb „Anwendungsobjekt“. In dieser „nicht fertigen“ Klasse wird AppKontext bereitgestellt. D.h. wird eine neue eigene Klasse benötigt, wird diese von Anwendungsobjekt abgeleitet und besitzt damit AppKontext.

Frage

* Warum ist im Infrastruktur-Objekt (im AppKontext) beim Debuggen die Liste mit den Sprachen enthalten, wenn ich doch nirgends diese Objekte initialisiere?

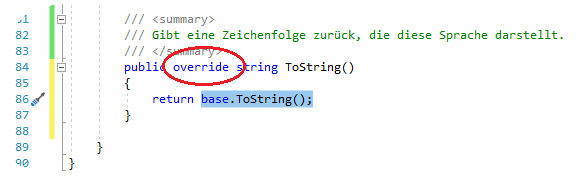
Weil wir gecachet programmieren und alle Objekte „just – in – time“ initialisiert werden. Beim Debuggen werden unsere Eigenschaften vom Visual Studio Debugger aufgerufen.

Zur Hausübung:

* Unterschied Klassen und Strukturen
  + Klassen sind Verweistypen, enthalten also nur eine Speicheradresse. Standardinitialisierung ist null
  + Strukturen sind Werttypen. Diese Variablen enthalten die Information direkt. Standard die Zahl 0 (Null)

Objektorientierte Programmierung mit Vererbung

* Immer wieder kommt vor, dass man  
  von einer Basisklasse etwas erbt, mit  
  dem man nicht glücklich ist



* In diesem Fall kann man sich über das Geerbte „hinwegsetzen“ (override)
* Wird auch gerne als „Überschreiben“ bezeichnet

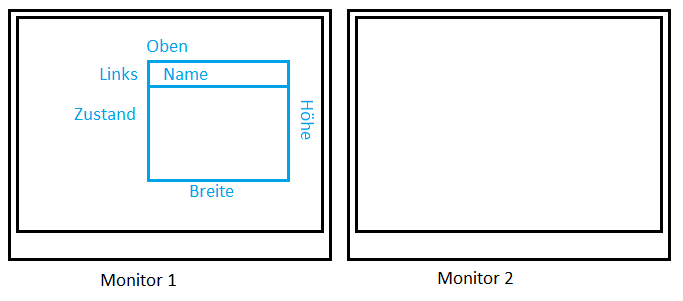
Das C# „override“ Schlüsselwort zeigt alle Mitglieder der Basisklasse, die „virtual“ modifiziert sind.

Hinweis:

* Die meisten Klassen überschreiben ToString()
* Nie auf das Ergebnis von ToString() verlassen
* Nur bei Werttypen bekommt man den Inhalt als Text.

Nächste Baustelle

Unsere Anwendungen sollen sich die Fensterpositionen   
und den Zustand merken.



Eigenschaften

* Name
* Zustand
* Links
* Oben
* Breite
* Höhe

Sicherstellen, dass bei mehreren Monitoren „unsere Fenster nicht verschwinden“, wenn ein Monitor aktuelle nicht angeschlossen ist

Die Eigenschaften Name und Zustand sind kein Problem, weil hier immer Daten vorhanden sind.

Anders sieht’s bei Links, Oben, Breite und Höhe aus. Falls das Fenster noch nicht vorhanden war oder es maximiert oder minimiert geschlossen wurde, sind die Werte ungültig. Wegen …

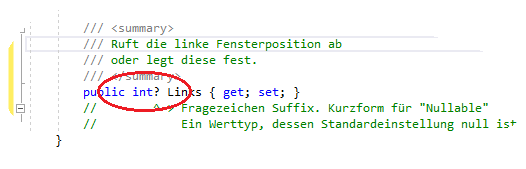
* Werttyp

… ist die Zahl 0 nicht interpretierbar.

Lösung: (seit 2002)

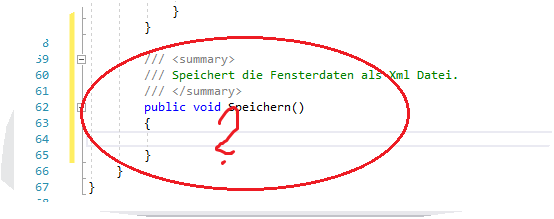
* Nullable Werttypen

<https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/nullable-types/>



Die Abbildung zeigt das Fragezeichen Suffix für nullable Werttypen.

Überlegungen zum Speichern



* Entweder den Benutzer fragen
* Wollen wir nicht für die Fensterpositionen
* AUF KEINEN FALL IM ANWENDUNGSVERZEICHNIS

Sie werden keinen Administrator finden, der ihnen das Programmverzeichnis aufmacht – Computer Viren

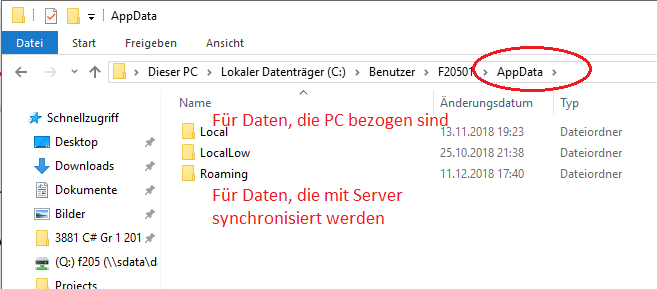
DIE ANWENDUNGSVERZEICHNISSE SIND TABU!!!!!!!!!

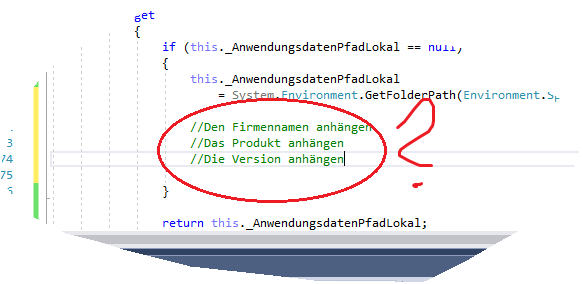
Wichtig:

* Falls ungefragt gespeichert wird, darf nur das   
  Benutzerprofil verwendet werden

Hier gibt’s einen versteckten Ordner für diese Zwecke:

* AppData



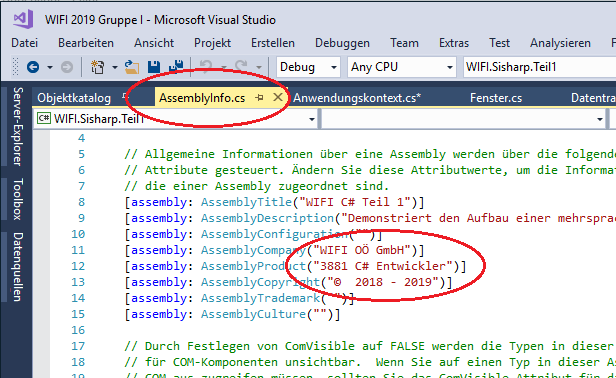


Wie kommen wir zum Firmenname, zum Produkt und der Version?

* Auf keinen Fall „hardcoded“ einfach hinschreiben

Wir haben das doch schon irgendwo hingeschrieben?

* In die AssemblyInfo



* Worum handelt es sich bei diesen [] eckigen Klammern Ausdrücken?

Um Attribute, so genannte Metadaten, die mitcompiliert werden und geholt werden können, wenn’s jemand braucht

* Wie heißt die Technik zum Auslesen von Informationen über ein Objekt zur Laufzeit?

Reflection

* Wo programmieren wir das hin?

Nicht einfach, weil in .Net gibt’s nur Einfachvererbung. Manchmal wird die Information benötigt, manchmal nicht. Kommt das in eine Basisklasse…

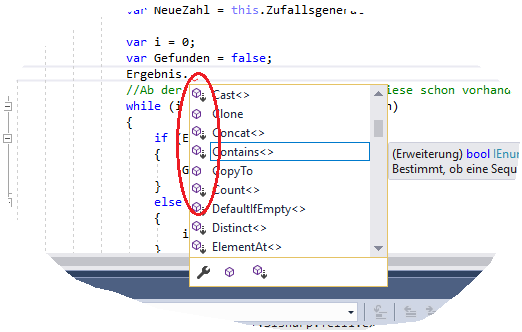
1. Eine überladene Basisklasse
2. Nicht alle unsere Objekte können dieselbe Basisklasse besitzen

* Was tun?

Das hat sich Microsoft auch gefragt…

Seit 2005

* Methoden, die man manchmal braucht, einem   
  Objekt „unterjubeln“
* Erweiterungsmethoden



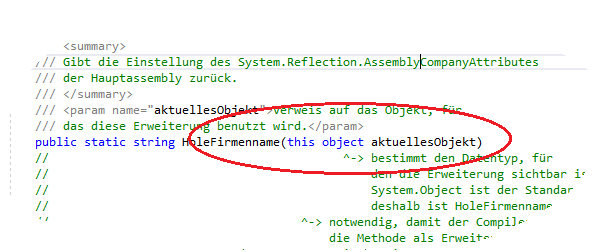
* Erkennbar an dem schwarzen Pfeil im Symbol
* Wir wollen so etwas auch

Hauptgrund für die Erfindung der Erweiterungen

* Language Integrated Query (LINQ)
* Abfragen auf Daten im Speicher ähnlich  
  der Structured Query Language (SQL)  
  für Abfragen auf Datenbanken

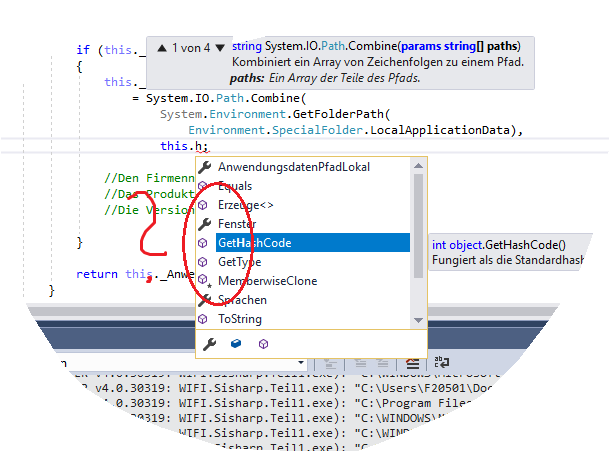
Handbuch: <https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/classes-and-structs/extension-methods>

Für eigene Erweiterungen …



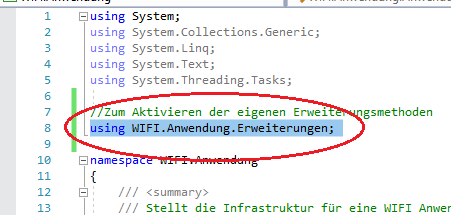
… statische Methoden, die in der Schnittstellte mit „this“ beginnen, implementieren.

Erweiterungen aktivieren

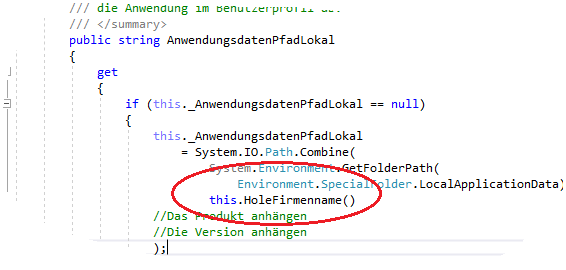


* Damit Erweiterungen aktiviert werden,  
  muss mit der „using“ Direktive der  
  Namespace angegeben werden, wo

die Erweiterungen zu finden sind



Erweiterungen …



… werden benutzt, als ob sie „geerbt“ worden wären.

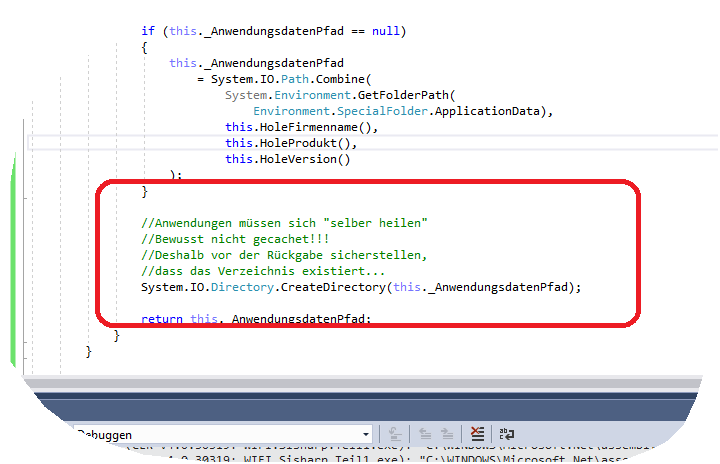
13.12.2018

Frage?

* Man kann doch die Erweiterungsmethode, z. B. WIFI.Anwendung.Erweiterungen.AssemblyInfo.HoleFirma() auch vollqualifiziert nutzen?  
    
  Warum dann diese Erweiterungen?  
    
  Das funktioniert bei unserem HoleFirma() nur zufällig, weil innerhalb der Methode nichts von dem Objekt, für das diese Erweiterung implementiert wurde, benötigt wird.
* Warum wird HoleFirma() nicht in der Basisklasse implementiert?  
    
  Weil wir nur Einfachvererbung haben und Anwendungskontext nicht die Basisklasse Anwendungsobjekt erweitert.
* Wir haben bereits XML erwähnt. Die selbstbeschreibende Textdatei, z. B. für unsere Sprachen. Jetzt habe ich XAML entdeckt – was ist das?  
    
  Hier handelt es sich um „eXtensible Application Markup Language“. Damit kein eine Oberfläche beschrieben und Klassen implizit initalisiert werden. Grundlage für Windows Presentation Foundation (WPF, Teil 2)

Tipp:

* Immer davon ausgehen, dass Benutzer irrtümlich etwas verschieben, löschen oder überhaupt ein neuer Benutzer vorhanden ist



* Anwendungen, die sich selber heilen, entwickeln.

Das Problem, wo ungefragt gespeichert werden darf, ist gelöst.

Zurück zum FensterManager.Speichern

Ständig gibt’s Code Teile, die …

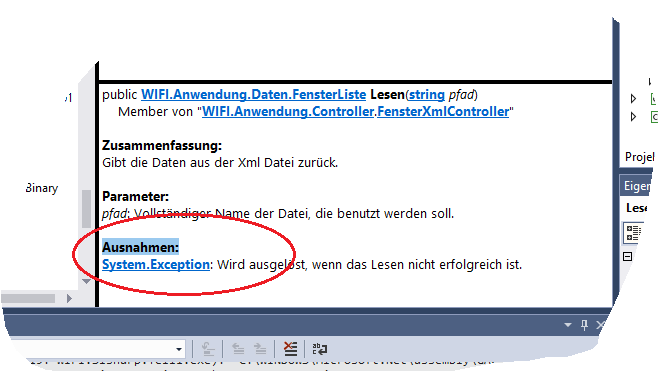
* Kritisch

… sind.

Was passiert bei kritischen Code-Teilen?

* Die Anwendung stürzt ab
* .Net kennt keine Fehlernummern
* .Net kennt nur Fehler-Objekte
* Alle Fehler-Objekt müssen  
    
  System.Exception  
    
  erweitern

Woran erkennt man kritischen Code?



* Wenn im Kommentar „Ausnahmen“ angeführt sind

Was tun mit kritischem Code?

* Fehlerbehandeln

C# Schlüssel für die Fehlerbehandlung?

* try-catch-finally

<https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/keywords/try-catch>

Beste Möglichkeit heute im catch-Zweig:

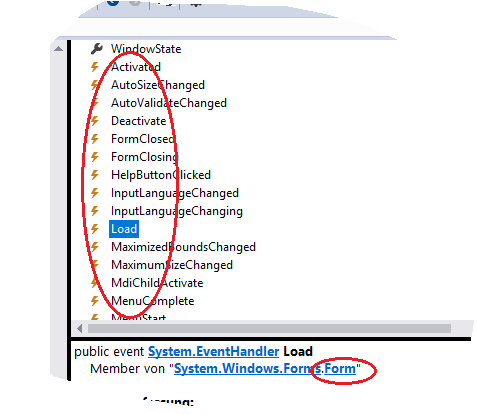
* Ein Ereignis auslösen, d.h. den  
  Benutzern die Möglichkeit geben,  
  eine eigene Methode anzuhängen

Wie funktioniert das technisch?

* Mit einem Spezialdatentyp, nichts anders als „string“ für Text, „int“ für Ganzzahlen
* Der Name von diesem Spezialdatentyp, der  
  die Speicheradresse einer Methode aufnehmen kann  
    
  delegate
* Entspricht den C-Funktionszeigern  
  Problem im C: Es ist alles erlaubt und man stürzt erst  
  recht wieder ab, wenn nicht sauber gearbeitet wird.
* Vorteil vom .Net Delegate  
  Er lässt nur Methoden zu, die der Delegate Deklaration entsprechen

Wozu wird den das jetzt benötigt?

* Weil wir beim Öffnen unseres Fenster die Fensterposition wiederherstellen müssen
* Microsoft hat beim Programmieren von System.Windows.Forms.Form keine Ahnung, was wir tun wollen
* Aber: Wir haben die Möglichkeit eine Methode „anzuhängen“. An was? An ein Ereignis



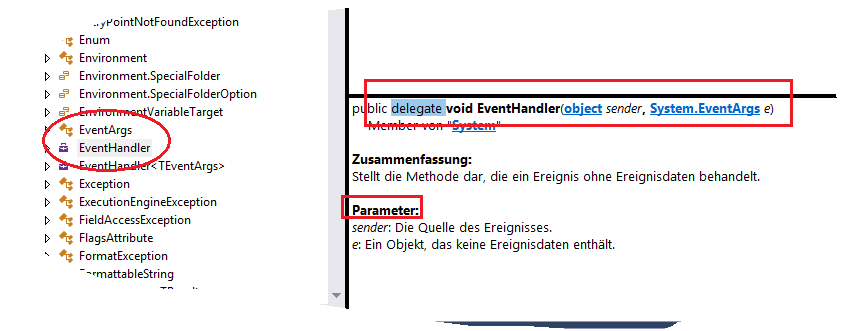
Bevor die Microsoft Ereignisse behandelt werden, ein eigenes Ereignis…

* FehlerAufgetreten

Schritte zum Implementieren eines Ereignisses:

1. Eine Klasse für die Daten, die mit dem Ereignis transportiert werden  
     
   Konvention:  
   Alle Ereignisdaten Klassen müssen System.EventArgs erweitern und im Namen auf „EventArgs“ enden
2. Einen Spezialdatentyp deklarieren, der eine Methode mit unseren Ereignisdaten aufnimmt (delegate)  
     
   Konvention:  
   Die Delegaten müssen im Namen auf „EventHandler“ enden und immer zwei Parameter besitzen…
   1. object sender
   2. EigeneEventArgs e

Wird ein Ereignis benötigt, das keine Zusatzdaten besitzt, kann Schritt 1 und 2 entfallen. Hier wird System.EventHandler mit System.EventArgs benutzt.

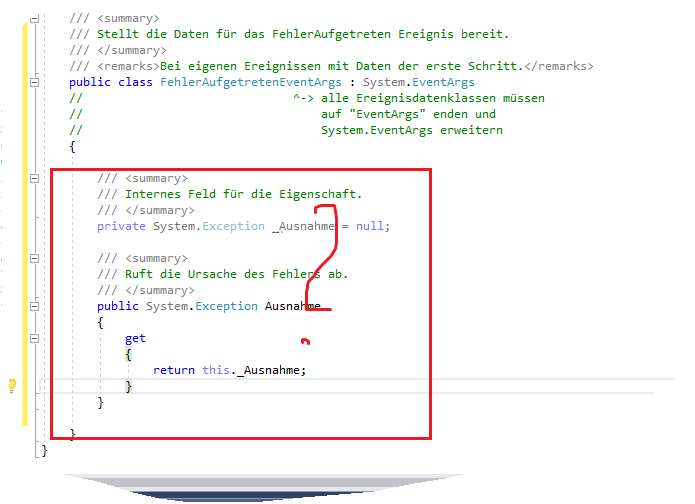


Wir wollen ein Ereignis, Konvention „ein verbaler Ausdruck“, wo die Ausnahme, warum der Fehler war, mittransportiert wird.

* Ereignis  
    
  FehlerAufgetreten

1. FehlerAufgetretenEventArgs

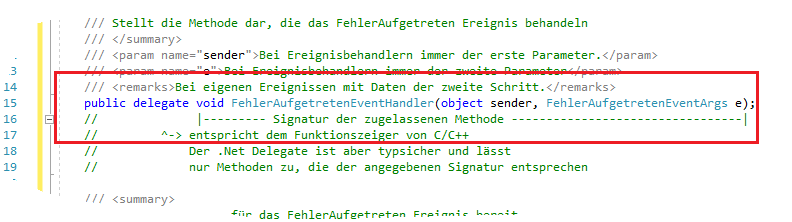
Wie bekommt man in ein Objekt einen Wert in eine …



… später schreibgeschützte Eigenschaft?

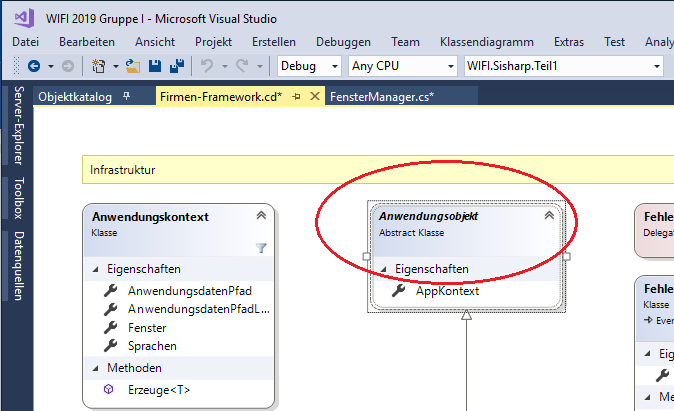
* Entweder, in dem sich das Objekt den Wert berechnet  
  Funktioniert hier nicht
* Oder über den Konstruktor

1. Ein Delegate für die neuen EventArgs



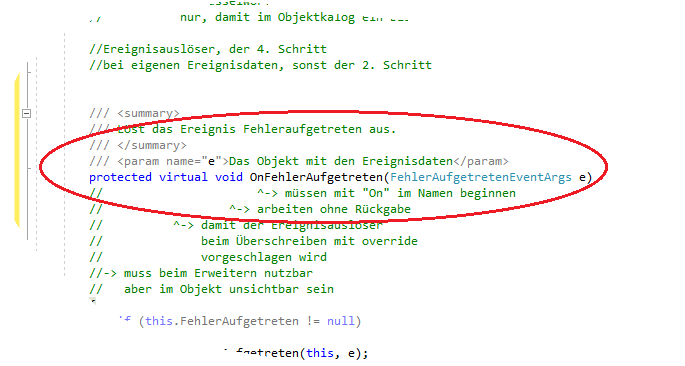
1. Das Ereignis deklarieren  
   (Punkt 1 und 2 können bei Ereignissen ohne Daten entfallen. In diesem Fall wird System.EventHandler benutzt)

Wo platzieren wir unser Ereignis?

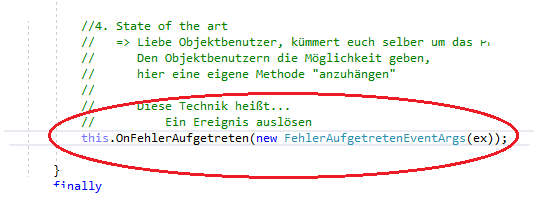


* Weil wir robuste Manager implementieren müssen und überall das Ereignis benötigt, im Anwendungsobjekt, weil alle Manager diese Klasse wegen der Infrastruktur erweitern müssen

1. Ein Ereignis ist ein Delegate, also die Möglichkeit  
   eine Methoden – Speicheradresse anzuhängen.  
   Keine Adresse ist null  
     
   Wir benötigten Methode, die prüft, ob eine  
   Speicheradresse vorhanden ist und wenn ja,  
   die Methode, den Ereignisbehandler, aufruft  
     
   Konvention:  
   Diese Ereignisauslöser müssen im Objekt unsichtbar sein,  
   beim Erweitern aber benutzt werden können (protected),  
   überschreibbar gekennzeichnet sein (virtual) und   
   im Namen mit „On“ beginnen und so heißen wie das   
   Ereignis



Anschließend kann überall dort, wo die angehängte Methode aufgerufen werden soll, …



… der Ereignisauslöser, d.h. die angehängte Methode vom Benutzer, aufgerufen werden.

* Das ist „Ereignis-objektorientierte Programmierung“

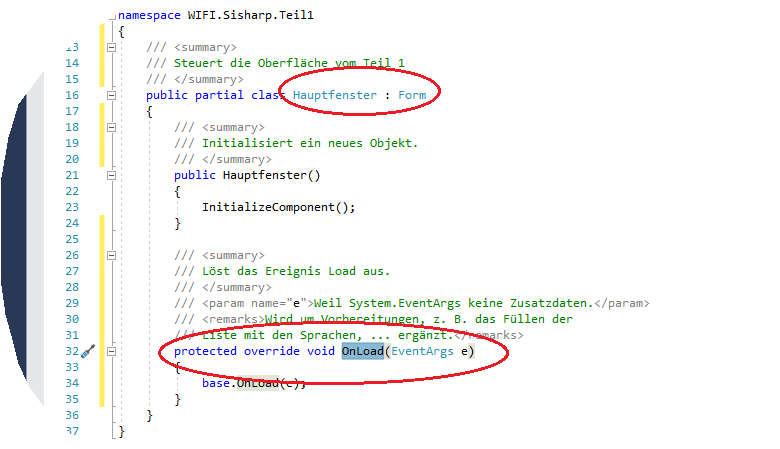
Offen ist, wie eine Speicheradresse einer Methode einem Delegate zugewiesen wird…

Zurück zum Teil 1

* Beim Öffnen des Fenster muss die Position wiederhergestellt werden

Unser „Hauptfenster“ ist eine Erweiterung von „System.Windows.Forms.Form“. Dieses Klasse löst ein Ereignis „Load“ aus und besitzt einen Ereignisauslöser „OnLoad“

Deshalb können wir, weil wir erweitern, das „Onload“ überschreiben



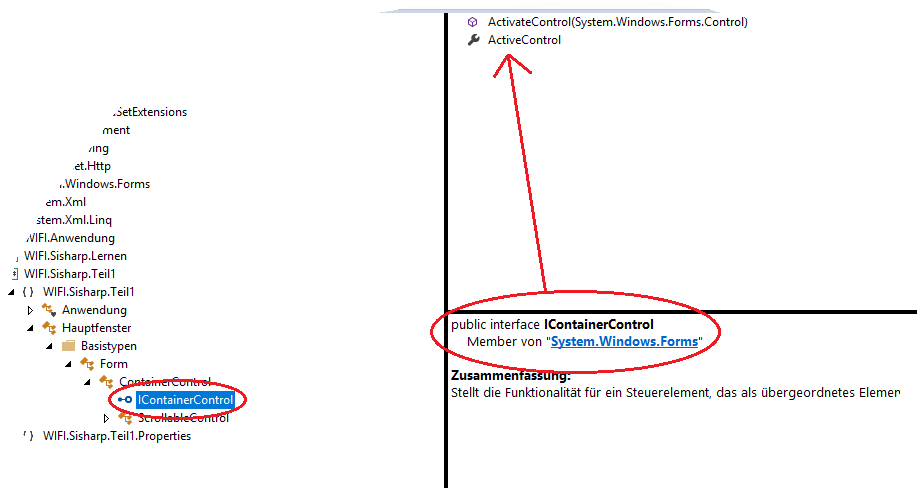
18.12.2018

Neues Problem:

* Es gibt nur Einfachvererbung
* Wir benötigten den Anwendungskontext überall
* Im Windows.Forms.Form Formular gibt’s den nicht
* Außerdem kann Anwendungskontext.Erzeuge<T>() nicht benutzt werden beim Formular, weil Erzeuge<T> nur Typen basierend auf Anwendungsobjekt zulässt.

Was tun?

* Eine Wunschliste schreiben an zukünftige Objekte schreiben
* Hier handelt es sich um „Interfaces“ (Schnittstellen, nicht mit den Schnittstellen einer Methode verwechseln)

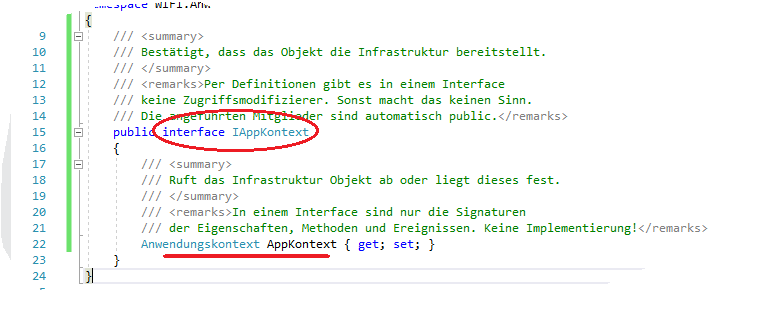


Die Abbildung zeigt, dass Microsoft die ScrollableControl Klasse zu ContainerControl erweitert hat. Von anderen Stellen gibt’s aber Wünsche an Klassen, die behaupten ein ContainerControl zu sein. Die Wünsche sind im Interface „IContainerControl“ zusammengefasst.

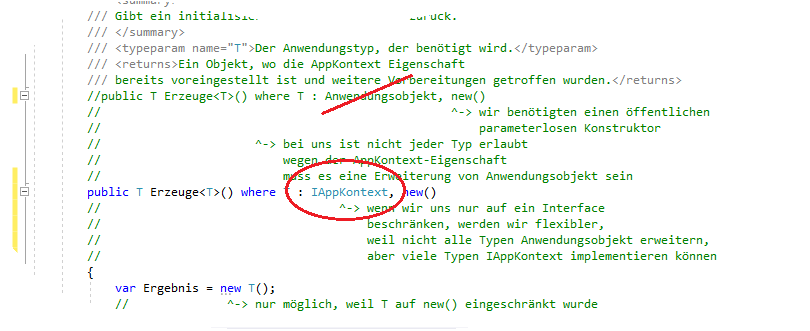
* Die Namen aller Interfaces müssen mit einem großen „i“ beginnen
* Eine Klasse kann nur eine Basisklasse haben, aber beliebig viele Interfaces implementieren (bestätigen)
* Ein Interface kann als Datentyp benutzt werden

Wir wollen ein Interface…

* IAppKontext, das sich die Eigenschaft AppKontext „wünscht“



Durch Benutzen von Interfaces …

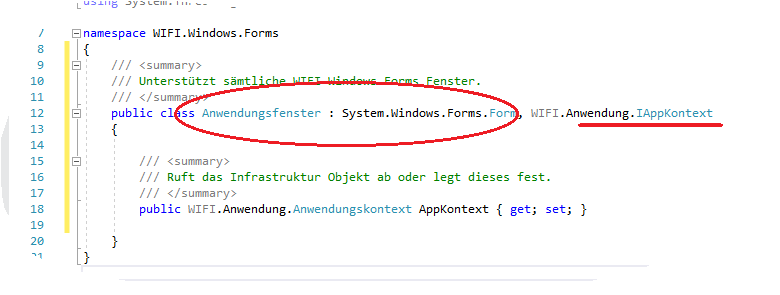


… bleiben wir bei steigender Flexibilität trotzdem „typsicher“

Programmierhandbuch:

<https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/interfaces/>

Vorteil beim Arbeiten mit Interfaces ist, dass …



… diese jetzt „überall“ benutzt werden können.

Komplett neue G’schichte:

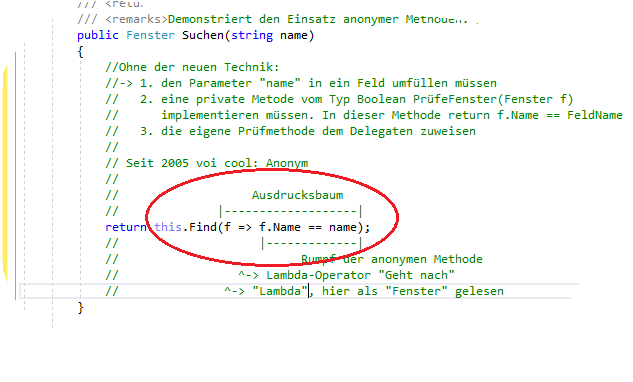
Bisher nur:

* Benannte Methoden
  + Ohne Rückgabe: void Arbeite(parameter)
  + Mit Rückgabe: Typ GibZurück(parameter)

Neu:

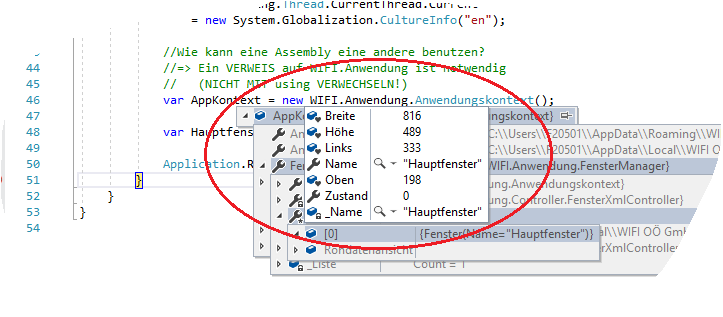
* Methoden ohne Name
* Anonyme Methoden, Lambda-Ausdrücke

<https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/statements-expressions-operators/anonymous-functions>



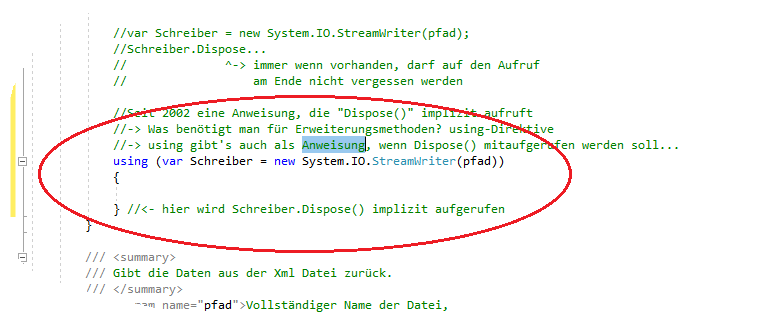
Die Abbildung zeigt, wie in der Fensterliste der Delegate für die Find Methode über eine anonyme Methode direkt zugewiesen wird.

Wie können Daten des Arbeitsspeichers auf einem Datenträger fixiert werden?



1. Mühsames selber programmieren
2. „Serialisierung“ von .Net benutzen  
     
   <https://docs.microsoft.com/de-de/dotnet/standard/serialization/>

Wie bereits früher erwähnt, darf man nie auf das Aufrufen der Dispose() Methode vergessen, wenn diese von einem Objekt bereitgestellt wird.



* Seit 2002 kann „using“ als Anweisung benutzt werden, wenn ein Objekt mit Dispose() benutzt wird

20.12.2018

Umsetzen vom Prototyp

Nach dem neuen Windows Forms Projekt

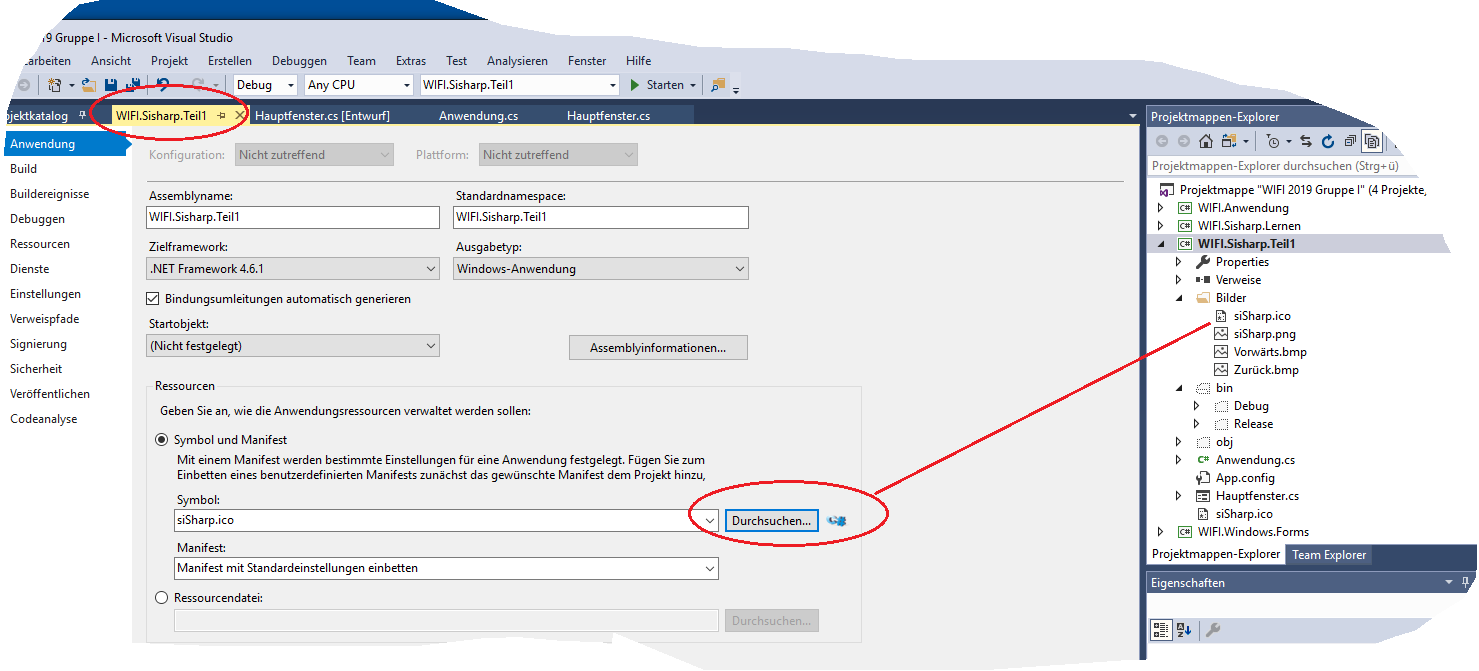
* Verweis auf WIFI.Anwendung (wegen der Infrastruktur)
* Verweis auf WIFI.Windows.Forms (damit wir ein Form bekommen, das sich die Fensterposition merkt
* Die AssemblyInfo sauber ausfüllen (wird von der Infrastruktur zum Berechnen von Speicherorten benutzt)
* Im Main
  + Die Infrastruktur WIFI.Anwendung.Anwendungskontext hochfahren
  + Alle zukünftigen Anwendungsobjekte mit der generischen Objektfabrik vom Anwendungskontext „Erzeuge<T>()“ initialisieren, damit der Anwendungskontext überall ist

Bevor man großartig mit der Oberfläche beginnt …

* Richtlinien lesen

<https://docs.microsoft.com/de-at/windows/desktop/>

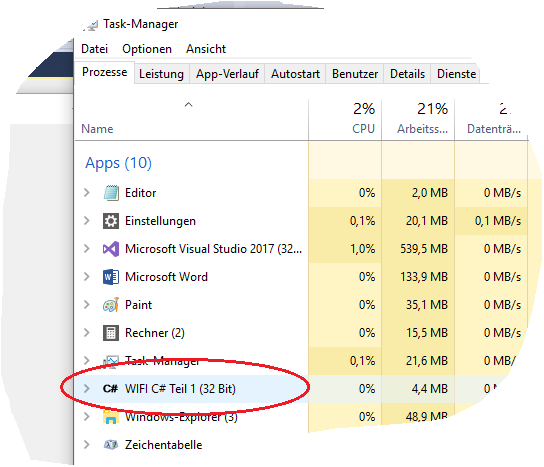
Dann …



… über die Projekteigenschaften eine Icon-Datei für die Anwendung aussuchen.

(<https://docs.microsoft.com/de-at/windows/desktop/uxguide/vis-icons>)

Problem:

* Unsere Anwendung läuft im 64 Bit Betriebssystem als…  
    
  … 32 Bit Anwendung

Wir hätten gerne 64 Bit…

Wichtig:

* Nur das 64 Bit Betriebssystem kann 32 Bit Anwendungen laufen lassen
* Innerhalb einer Anwendung KANN NICHT GEMISCHT werden!
* Entscheidet man sich für 64 Bit, müssen alle benutzten Komponenten ebenfalls als 64 Bit Teile vorlagen. Das ist bei Drittanbietern oft nicht der Fall
* Deshalb ist von Microsoft beim Office dringend empfohlen, die 32 Bit Version zu installieren
* Unterschied:
  + 32 Bit 2 Hoch 32 ergibt ca 4 Gigabyte Speicher, der von einer 32 Bit Anwendung angesprochen
  + 64 Bit -> 2 Hoch 64 (selber ausrechnen)

Wie kommen wir zu 64 Bit?

* Wie ist das .Net aufgebaut?
  + Auf erster Ebene sind die   
      
    „Assemblies“
  + Was ist eine Assembly?

Der Prozessor versteht nur 0 und 1 (binär)

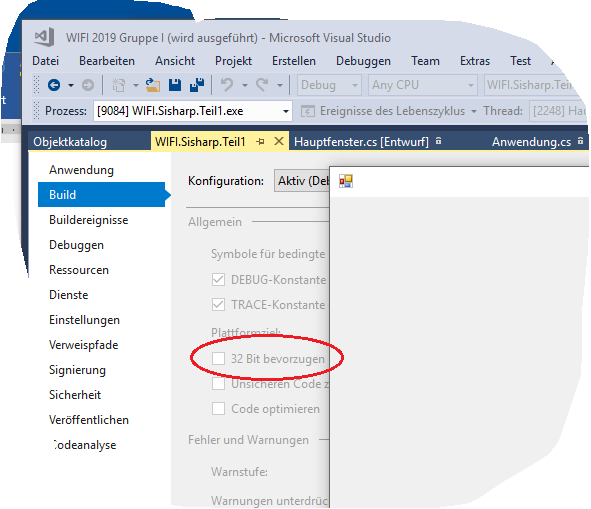
1. Generation der Programmierung  
     
   Binärcode, z. B. 0010 1101  
     
   War furchtbar fad, Löcher auf einer Lochkarte machen
2. Generation  
     
   Assembler  
     
   Hier wurden Bit-Kombinationen „Kürzel“ verpasst,   
   z. B. ADD für 0010 1101  
     
   Neues Problem:  
   Der Prozessor versteht das nicht mehr.  
     
   Ab der 2. Generation wird ein Zusatzprogramm benötigt, der den „Programmcode“ wieder in „Binärcode“ übersetzt.  
     
   => Compiler  
   (Wird für jeden Prozessor extra benötigt)
3. Generation  
     
   „Höheren Programmiersprachen“  
   angelehnt an eine natürliche Sprache.  
     
   Perfekt mit der natürlichen Sprache: Cobol. Hier wurde ganze Sätze geschrieben. Hier hört auch eine Anweisung mit einem Punkt auf.  
     
   Beispiele  
   BASIC, C, Fortran, C/C++, ….  
   und C# (mit .Net auf den Markt gebracht)  
   JavaScript
4. Generation  
     
   „Grafische Programmieren“.  
   Hauptsächlich in der Maschinenprogrammierung (Automatisieren)

Wie funktionieren .Net Compiler?

* Die .Net Compiler sind eine Ausnahme
* Die machen keinen Maschinencode
* Diese erzeugen nur Assembler – Code (Microsoft Intermediat Code)  
  (Deshalb auch Assemblies)
* Dieser Assembler wird „Just – In – Time“ beim Ausführen   
  „fertig compiliert“. D. h. der Kern der .Net Umgebung ist der „JIT“,  
  der Just-In-Time Compiler  
  Der ist aber relativ schlank und steht für jeden Prozessor und für 32 Bit und 64 Bit bereit.

<https://docs.microsoft.com/de-de/dotnet/framework/get-started/overview>

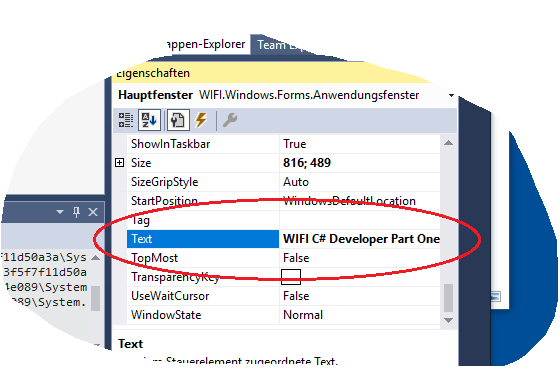
Beim Entwickeln (ohne Fremdkomponenten) muss deshalb nicht mehr zwischen 32 Bit und 64 Bit unterschieden werden. Man kann nur den benutzten JIT Compiler einschränken.



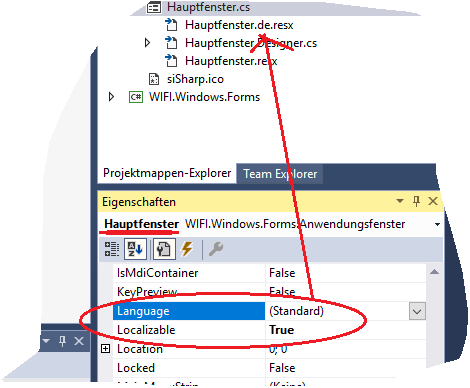
Dazu die Projekteigenschaften / Seite Build benutzen

Zu den mehrsprachigen Anwendungen:

* In unserm Fall Deutsch und Englisch
* Welche Sprache wird beim Entwickeln der  
  Oberfläche benutzt?  
    
  Die Sprache, die benutzt werden soll,  
  wenn es keine Übersetzung gibt  
    
  Heute im Normalfall Englisch



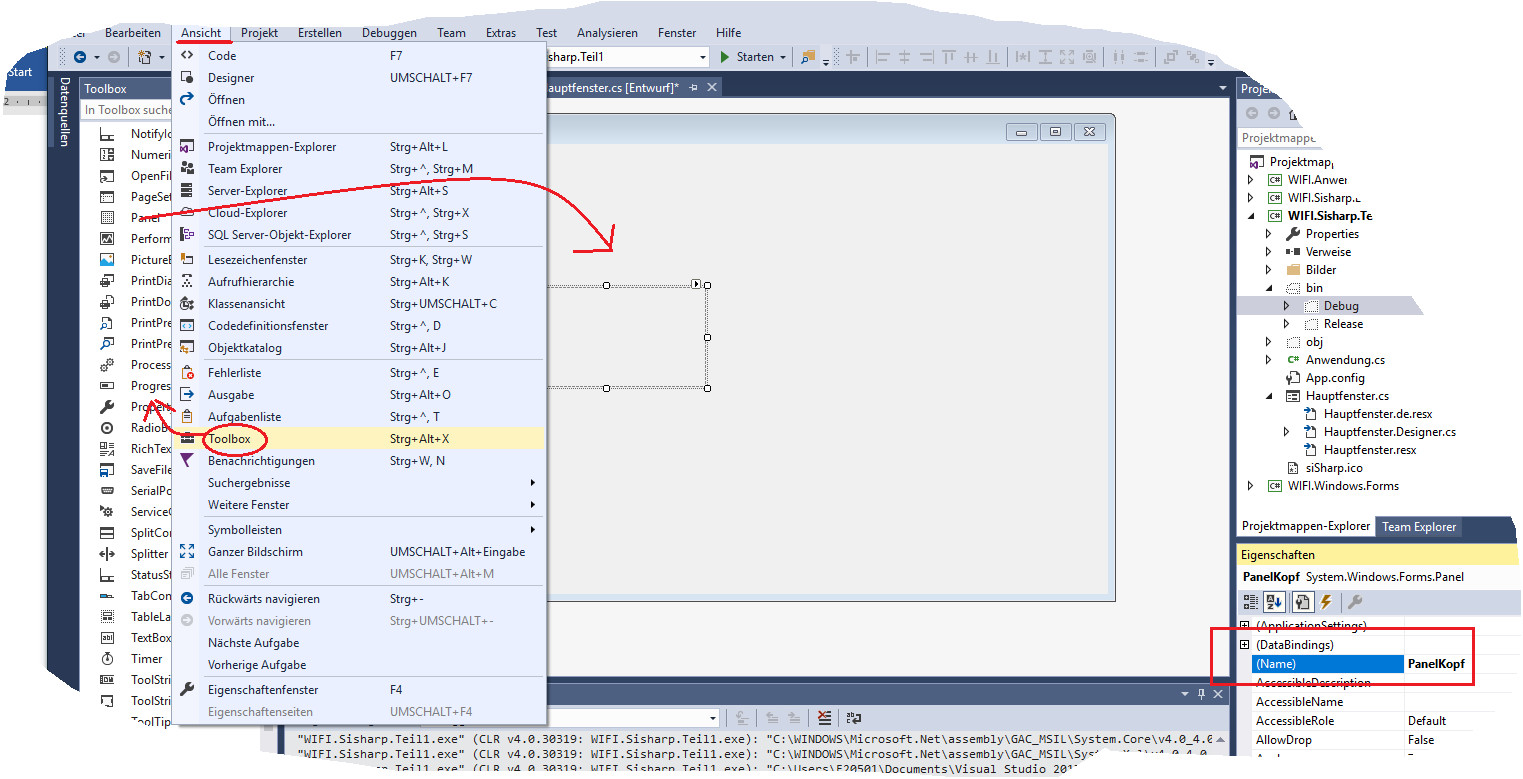
* Wird ein Windows Form Fenster übersetzt, d.h. lokalisiert?



1. Die „Localizable“ Eigenschaft des Formulars auf True
2. Die Sprache „Language“, z. B. auf „Deutsch“ ändern
3. Die Texte in den Eigenschaften ändern
4. Anschließend sofort wieder die Sprache auf „(Standard)“ ändern, weil sonst Änderungen nur in dieser Sprache, aber nicht in der Basissprache geändert werden.

Grundsätzlich dieselbe Technik, die bereits in WIFI.Anwendung für die lokalisierten Sprachen benutzt wurde.

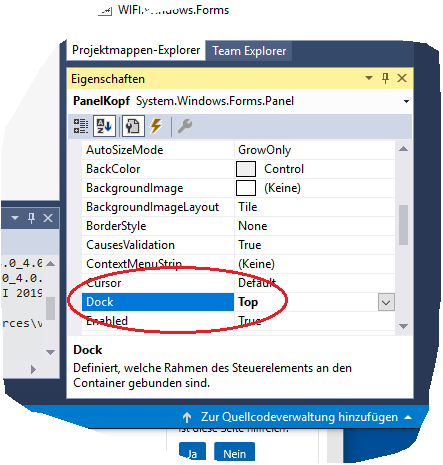
Designen der Oberfläche



* Visual Studio stellt hier einen grafischen Designer bereit.
* Aus der ToolBox benötigte Steuerelemente in das Fenster ziehen
* SOFORT jedes neue Steuerelement benennen!  
  Studio erstellt im DesignerCode für jedes gezeichnete Steuerelement ein Feld.

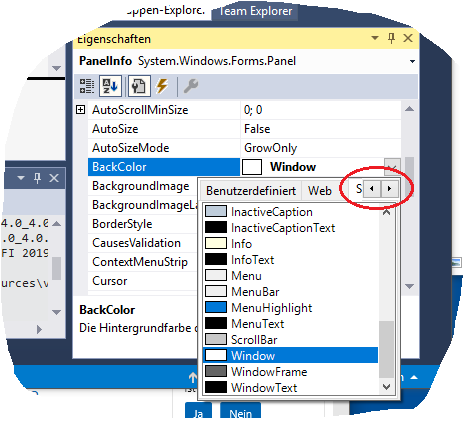
Richtlinien zu den Steuerelementen: <https://docs.microsoft.com/de-at/windows/desktop/uxguide/controls>

Die bei Hauptfenster Steuerelementen geänderten Eigenschaften werden im Eigenschaftenfenster …



… **fett** dargestellt.

Tipp zu den Farben:

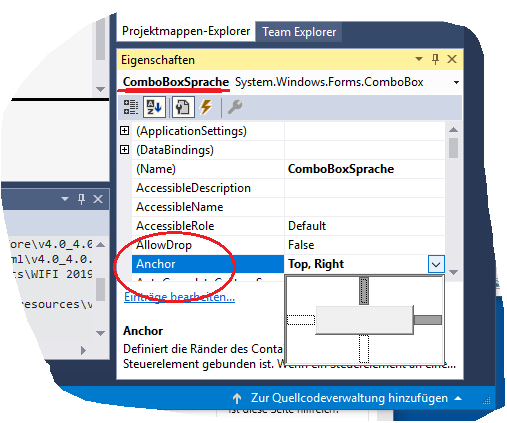


* Am besten aus dem System benutzen, damit die eigene Anwendung auf Farbänderungen des Benutzers im Betriebssystem reagiert

In den Richtlinien zum Lokalisieren heißt’s:

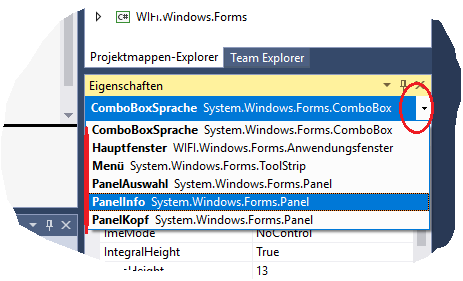
* Dem Benutzer die Möglichkeit geben, die Sprache auszuwählen
* „Stellen Sie sich vor, Sie sitzen in einem chinesischen Internetcafé und könnten die Sprache nicht einstellen“

Praktisch seit .Net Windows Forms



* Die Anchor Eigenschaft.
* Mit dieser kann erreicht werden, dass sich Objekte beim Ändern der Fenstergröße automatisch mitbewegen

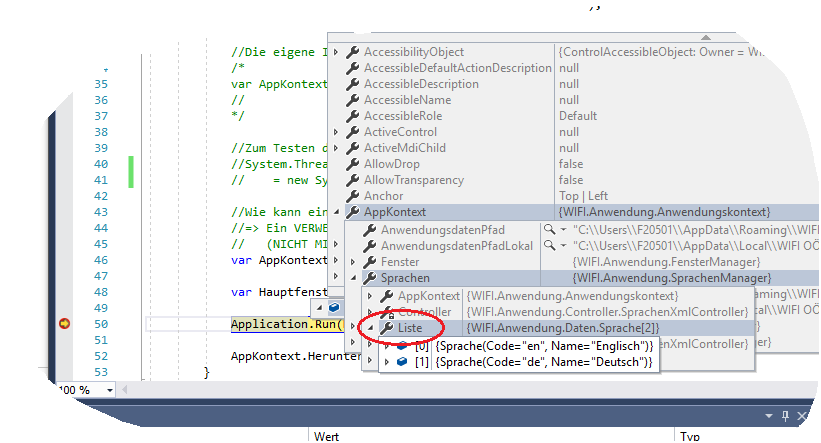
Hilfe, ich habe Steuerelemente „verloren“. Was tun?



* Im Eigenschaften Panel die Liste öffnen und vermeintliche „verschwundene“ Steuerelement markieren

Damit wir auch ein Bisschen programmieren…

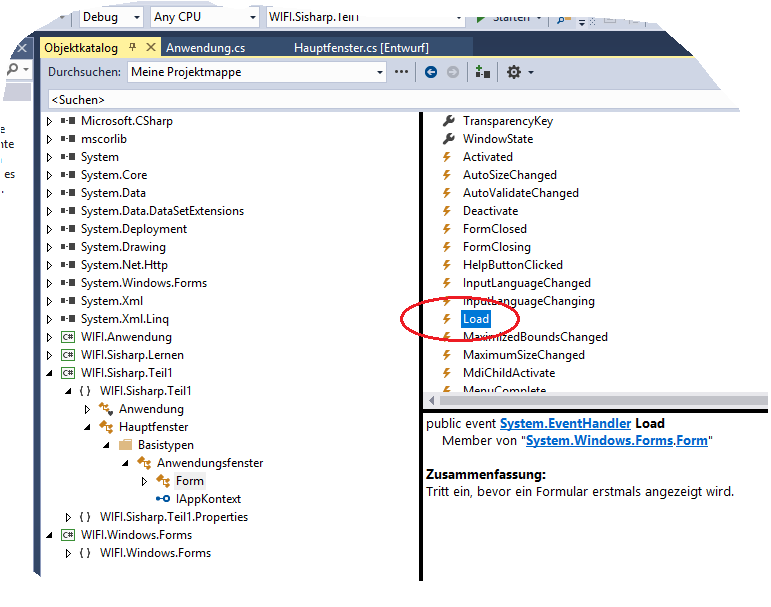
* Die Liste mit den Sprachen füllen



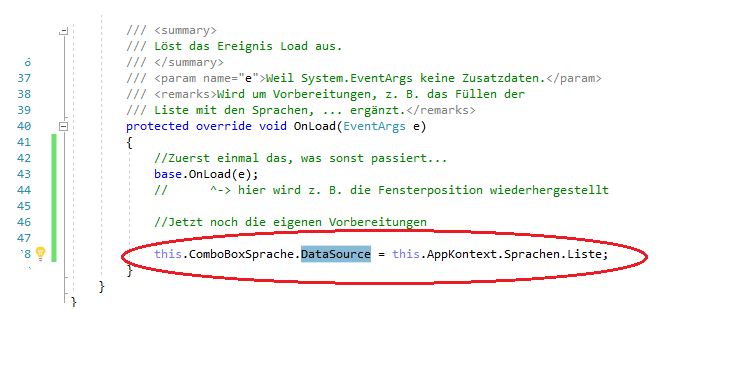
Die Abbildung zeigt, dass das Hauptfenster die Sprachen bereits im Infrastrukturobjekt „AppKontext“ hat.

Wo „tippen“ wir das Füllen der Liste?

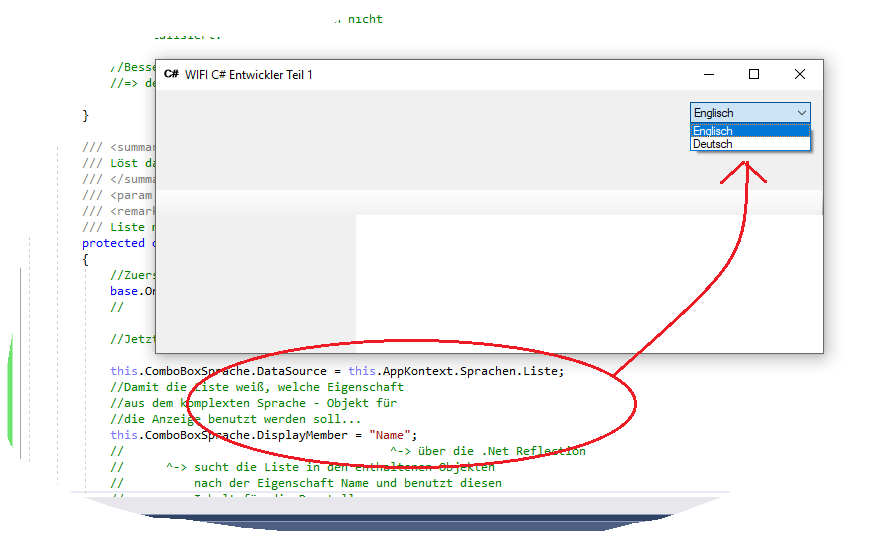
* Wenn das Formular geöffnet wird
* NICHT im Konstruktor, weil hier die Eigenschaften noch nicht bereitstehen
* Wir benötigen ein Ereignis



* Es bietet sich wie beim Anwendungsfenster des Frameworks das Load Ereignis an
  + Ein Ereignis ist ein Delegate, d.h. die Möglichkeit, eine Programmspeicheradresse anzuhängen
  + Jedes Ereignis hat einen Ereignisauslöser
  + Der Ereignisauslöser muss im Namen mit „On“ beginnen
  + Wird eine Klasse erweitert, keine Ereignisbehandler-Methode an das Ereignis anhängen
  + Besser  
      
    Den Ereignisauslöser überschreiben



Die Abbildung zeigt, wie aus der Infrastruktur die Liste mit den Sprachen an die Liste über die DataSource Eigenschaft „gebunden“ werden.



Wo merken wir uns die Sprache, die die Benutzer zuletzt eingestellt hatten?

* In der .Net Anwendungskonfiguration

Windows 1 bis 3.0

* In INI-Dateien

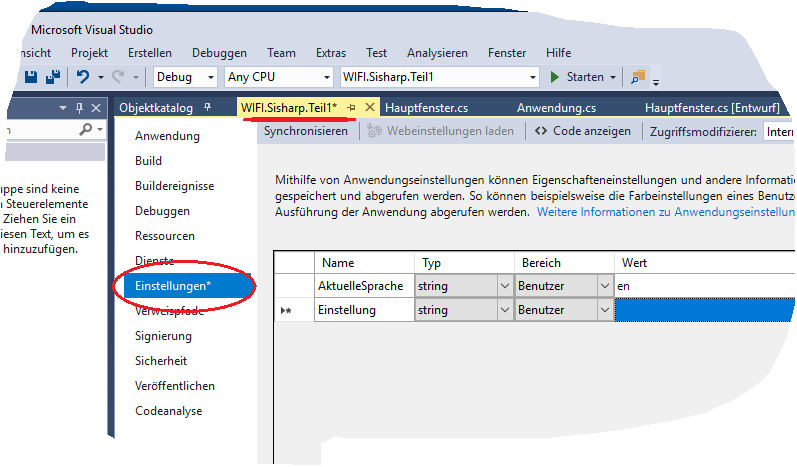
Windows 3.1 bis XP

* In der Registry

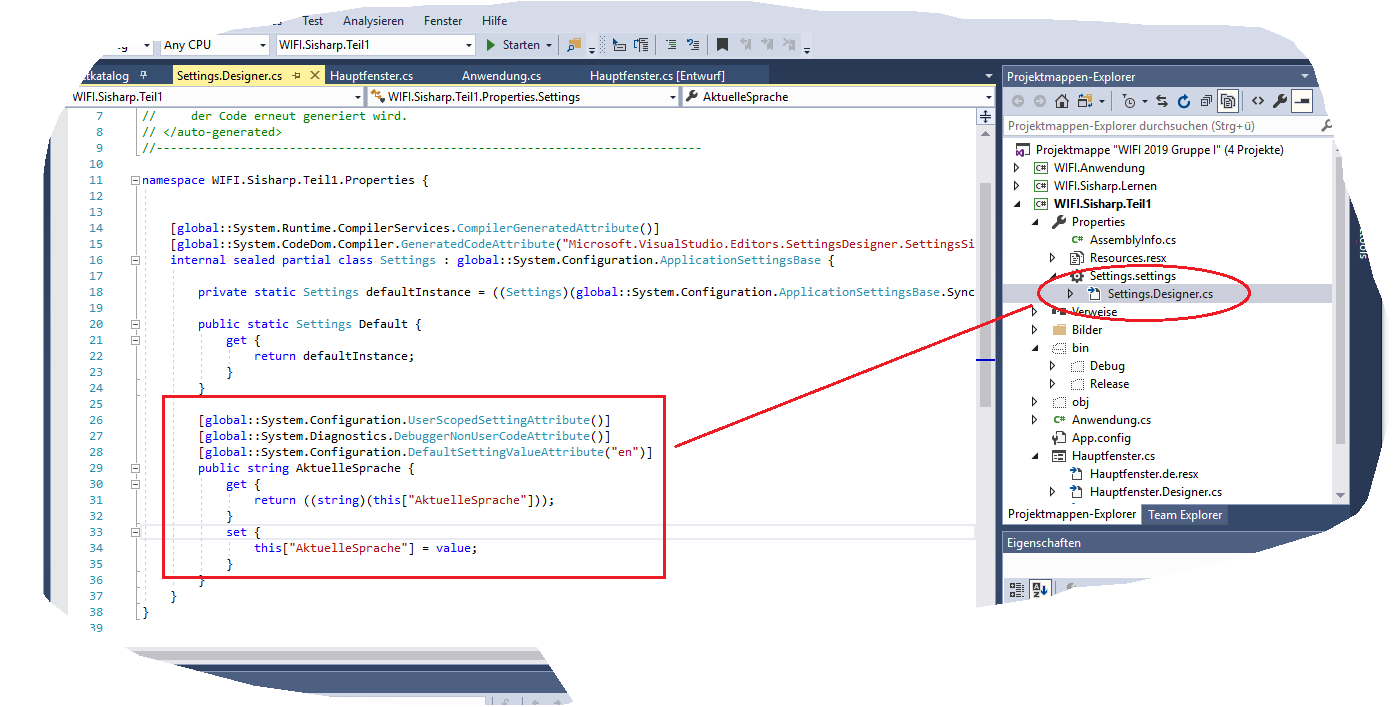
Seit .Net

* In Xml-Konfigurationsdateien
  + Domänen Konfiguration (von den Administratoren)
  + Computer Konfiguration (von den Administratoren)
  + Anwendungskonfiguration (für uns Tipper)

Die Anwendungskonfiguration …



… kann über die Eigenschaften / Einstellungen verwaltet werden.



* Das Studio erstellt automatisch eine Klasse „Settings“, wo die Einstellungen der Konfiguration als Eigenschaften bereitgestellt werden

Bevor hier weitergemacht werden kann, muss der SprachenManager des Firmenframeworks ebenfalls die aktuelle Sprache beherrschen.

Wie heißt die Technik, um auf Daten in einer Datenbank zuzugreifen?

* Structured Query Language (SQL, Teil 2)

Seit 2005 kann aber auch .Net ähnlich wie SQL auf Daten zugreifen, die sich im Arbeitsspeicher befinden:

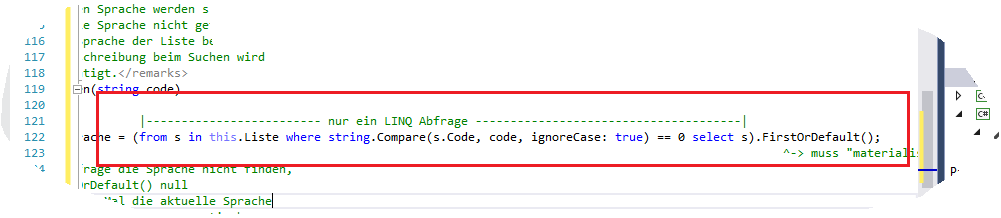
* Sprachintegrierte Abfragetechnik (LINQ)

Hauptschlüsselwort von SQL:

* SELECT  
  Kommt bei LINQ zum Schluss
* FROM  
  Mit dem beginnt LINQ

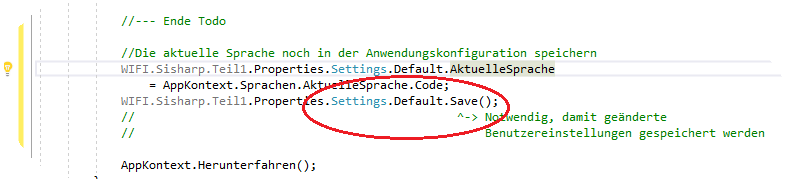
<https://docs.microsoft.com/de-de/dotnet/standard/using-linq>

<https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/concepts/linq/>



Die Abbildung zeigt, wie mit LINQ die Sprache für einen bestimmten Code ermittelt wird.

Zurück zur Anwendungskonfiguration:

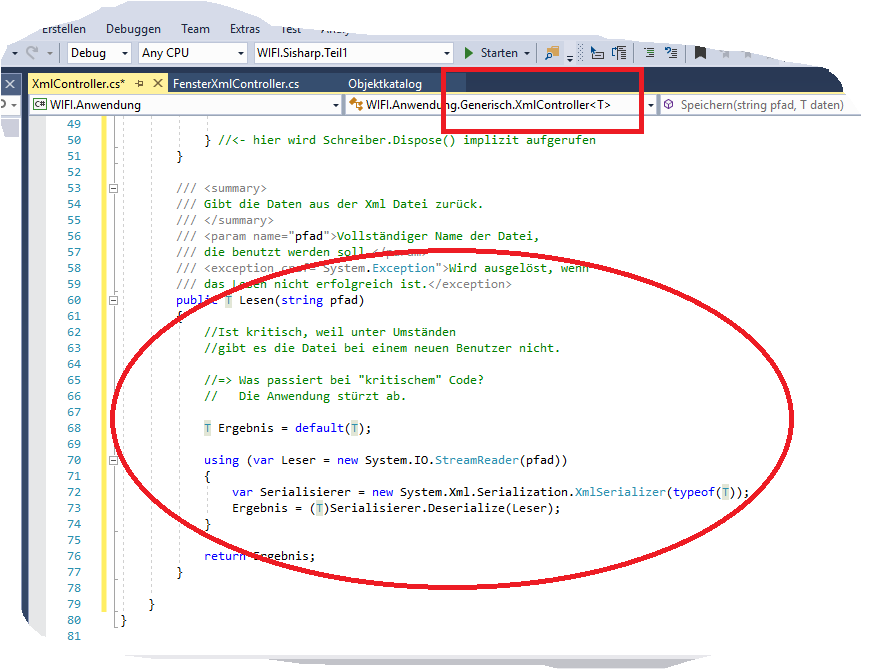
* Die muss für Benutzereinstellungen…  
    
  … ausdrücklich gespeichert werden

17. Jänner 2019

Hinweis:

* Es gibt eine Betriebssystem Batchdatei „Clean.cmd“, mit der alle ausführbaren Dateien aus einer Studio Projektmappe gelöscht werden können, damit die Mappe als Zip über z. B. gmail Konten verschickt werden kann.

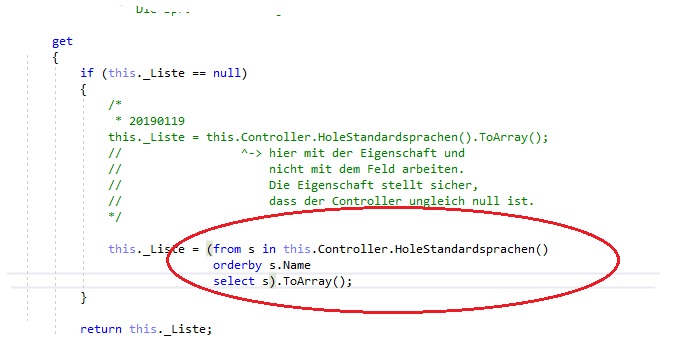
Klassen sind einfach von einem speziellen Typ auf „generisch“ umstellbar…



* Die Abbildung zeigt, wie der FensterXmlController zu einer generischen XmlController<T> Klasse umgebaut wird

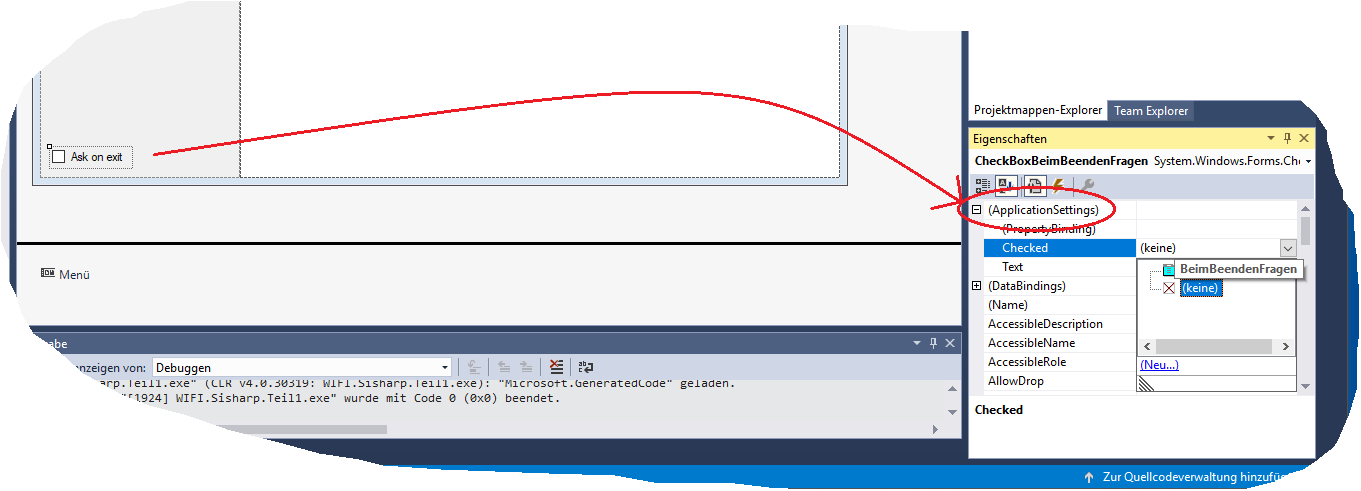
Voi, voi praktisch:

* Abfragen von Daten im Speicher mit LINQ



Die Abbildung zeigt, wie die Sprachen, die vom Controller irgendwie zurückgegeben werden, vom Manager nach Name sortiert präsentiert werden.

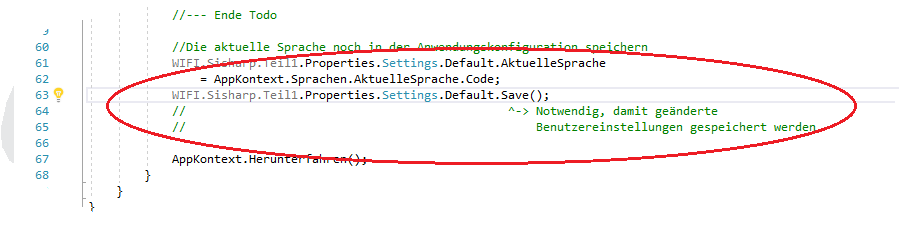
Praktisch:



* Bestimmte Eigenschaften, z. B. „Checked“ bei einem Kontrollkästchen kann an die Anwendungskonfiguration „gebunden“ werden.

Wichtig:

* Nicht auf das Speichern der Einstellungen der Anwendungskonfiguration vergessen!



Problem:

* Warum sind die Sprachen in deutscher Sprache, obwohl Englisch läuft?

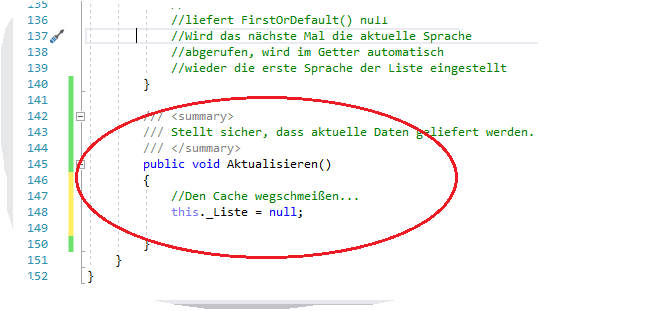
Das kann nur ein Problem mit dem Cache sein.

Warum?

* Unser Betriebssystem ist deutsch
* D.h. beim ersten Zugriff auf den Sprachmanager holt dieser die Daten deutsch
* Dann wird im Thread die Sprache auf Englisch geändert
* Und die Liste mit den Sprachen angezeigt. Aber falsch

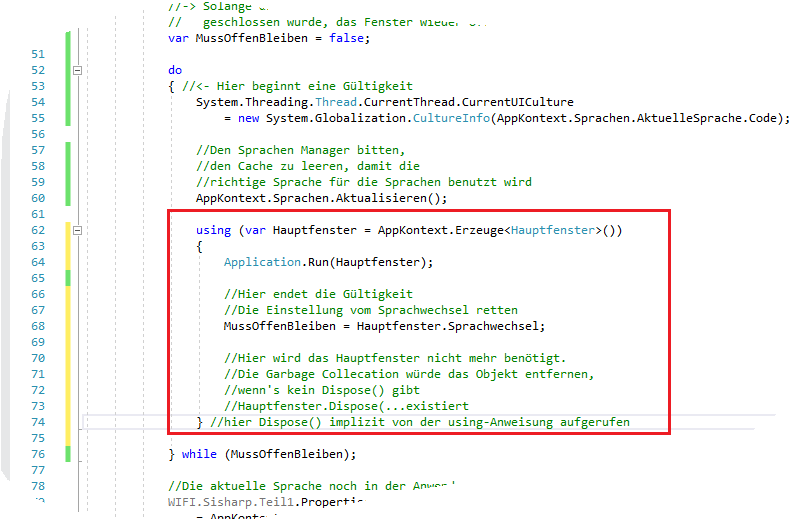
Lösung:

* Zwischen dem Wechseln der Sprache im Thread und
* Dem ersten Anzeigen
* Muss der Cache geleert werden, damit Sprachen Manager die Sprachen neu liest



In vielen Fällen reicht es zum Aktualisieren einfach den Cache zu entfernen. Durch unsere Programmiertechnik „reparieren“ sich die Eigenschaften, wenn das Feld null ist.

Wichtig:

* Auf keinen Fall …  
  

… auf das Aufrufen von Dispose() vergessen, wenn ein Objekt den Gültigkeitsbereich verlässt.

Wie kann bei Objekten auf ein Ereignis reagieren?

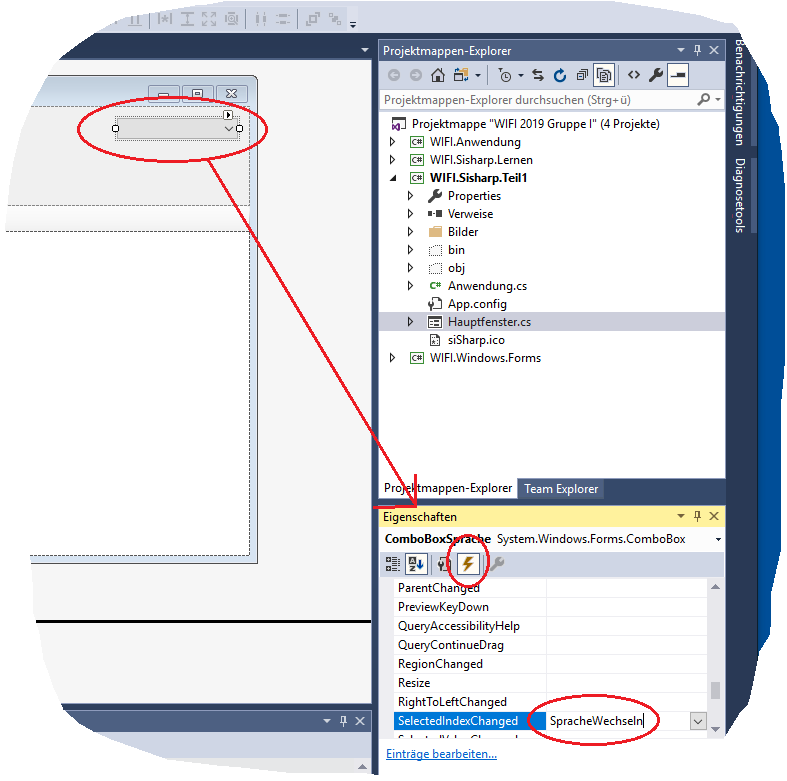
(Nicht verwechseln beim Erstellen einer Klasse. Hier wird der Ereignisauslöser „On…“ überschrieben)

Beim Wechseln der Sprache in der Liste, muss das Fenster geschlossen werden…

* Die Liste ist aber keine neue Klasse, die Liste ist ein Objekt
* Deshalb können wir nur auf ein Ereignis reagieren. Einen Ereignisbehandler anhängen

Erste Möglichkeit:

* In der Oberfläche



1. Objekt markieren
2. In den Eigenschaften auf Ereignisse wechseln
3. Gewünschtes Ereignis suchen  
   (Wie weiß man, welches? Gar ned, Erfahrung, probieren)
4. Den Namen des Behandlers tippen
5. Eingabe

Problem:

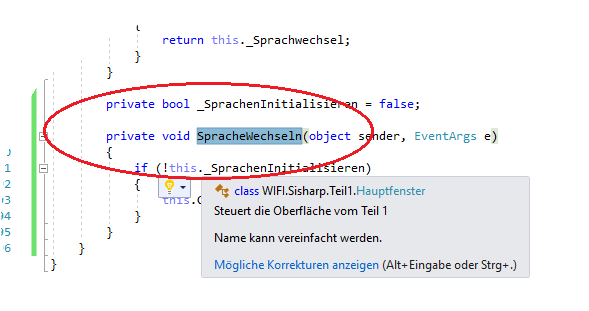
* Das SelectedIndexChanged Ereignis wird bereits ausgelöst,   
  wenn die Liste gefüllt wird. Weil aber das Fenster beim Wechseln der  
  Sprache geschlossen werden, funktioniert das Anhängen des Ereignis über die Oberfläche nicht so einfach.  
    
  Hier wäre ein zusätzliches Hilfsfeld notwendig.  
  Interessiert uns null

Zweite Möglichkeit:

* Direkt im Code bewusst einen Behandler anhängen

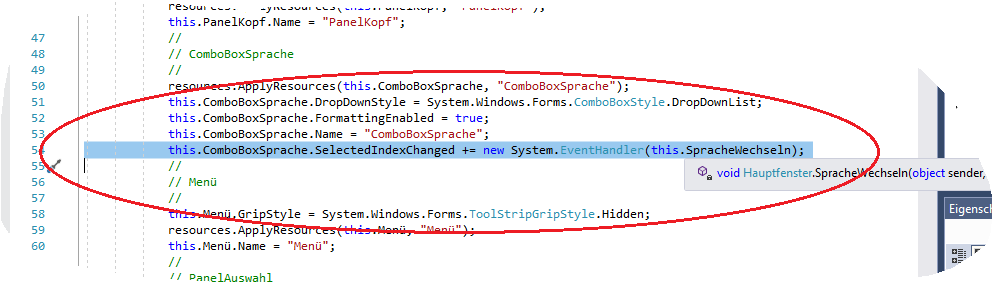
Weil:

* Zur ersten Möglichkeit zurück:



* Wo ist die Verknüpfung „SpracheWechseln“ mit der Liste

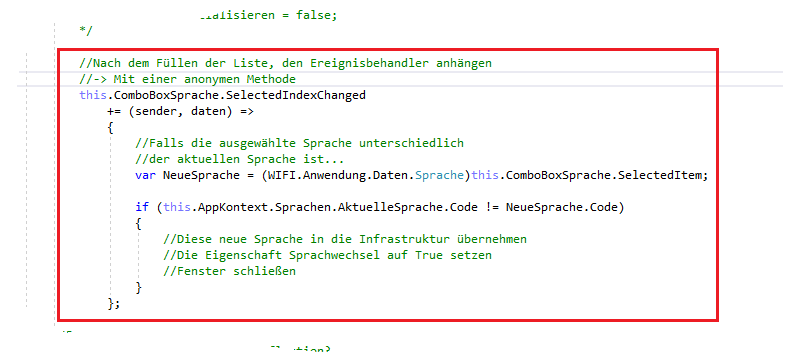
Diese ist im Code Behind, vom Studio



* Das Anhängen können wir selber auch
* Wir hängen den Ereignisbehandler aber erst an,  
  wenn die Liste gefüllt ist.  
  Damit brauchen wir kein Hilfsfeld

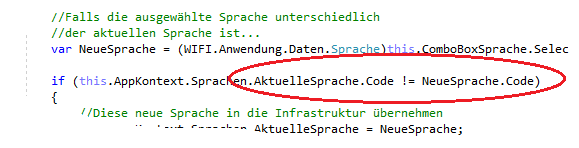
Wichtig:

* Keine Doppelklicks in der Oberfläche, weil Studio automatische Ereignisbehandler für das Standard-Ereignis erzeugt und im Code Behind die Behandler anhängt



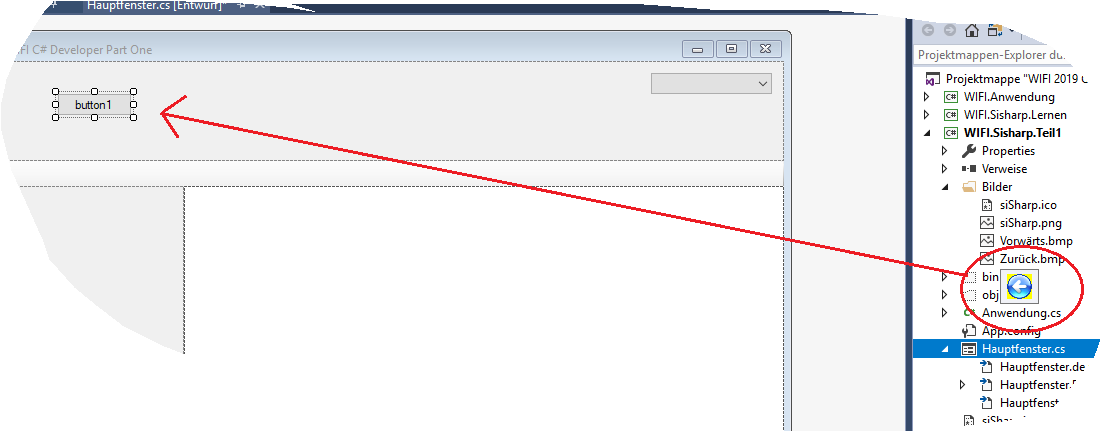
Die Abbildung zeigt, wie im OnLoad Ereignisauslöser Überschreiber dynamisch nach dem Füllen der Liste an das SelectedIndexChanged Ereignis ein anonymer Behandler angehängt wird.

Warum wird nicht „AktuelleSprache != NeueSprache“ getippt?



* Weil Sprache ein komplexes Objekt ist.   
  „AktuelleSprache == NeueSprache“ überprüft nur, ob die zwei Variablen im Arbeitsspeicher auf dieselbe Adresse zeigen.  
    
  Es muss etwas in den Objekten verglichen, was eindeutig ist. In unserem Fall ist das die Eigenschaft „Sprache.Code“

Zurück zur Oberfläche:



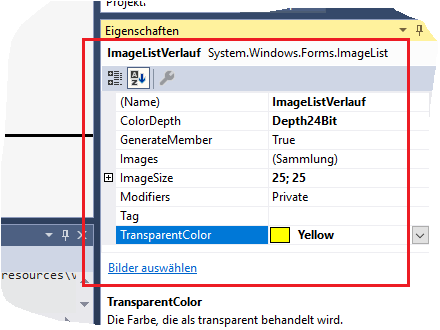
* Wie schaffen wir „transparente“ Schaltflächen

Windows Forms kann bei den Schaltflächen keine Transparanz.

Wird diese benötigt, ist eine …

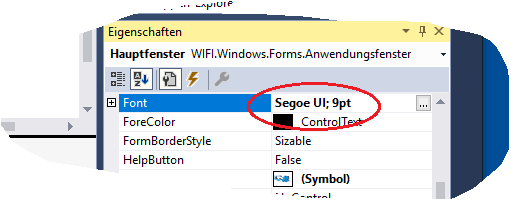
* ImageList

… notwendig.

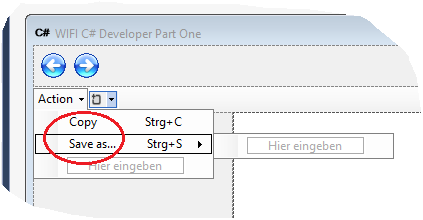


Bei der ImageList kann die Farbtiefe, die Größe und die Farbe, die Transparent werden soll, eingestellt werden.

Standardschrift für Windows Oberflächen:



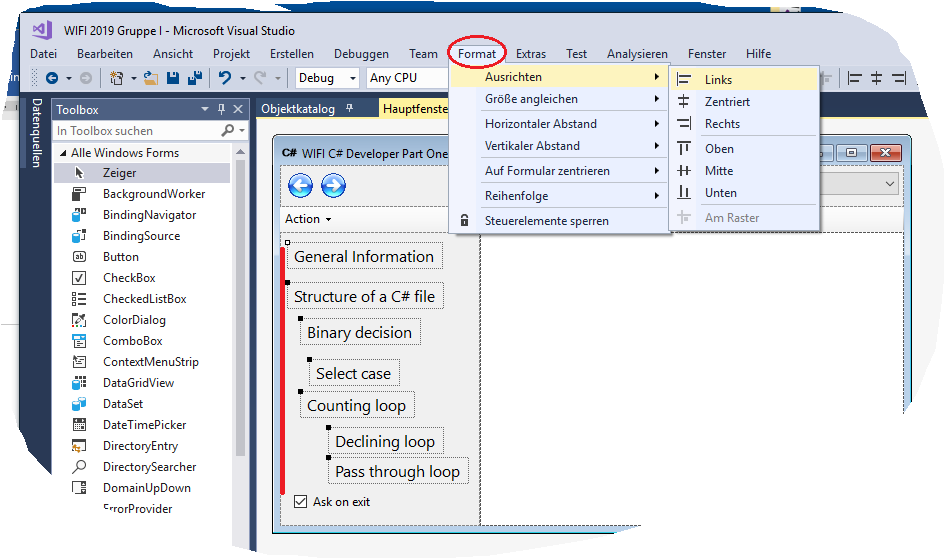
Aufpassen:



* Wenn ein Befehl ein weiteres Dialogfenster öffnet,   
  sind drei Punkte am Ende des Namens zu schreiben!

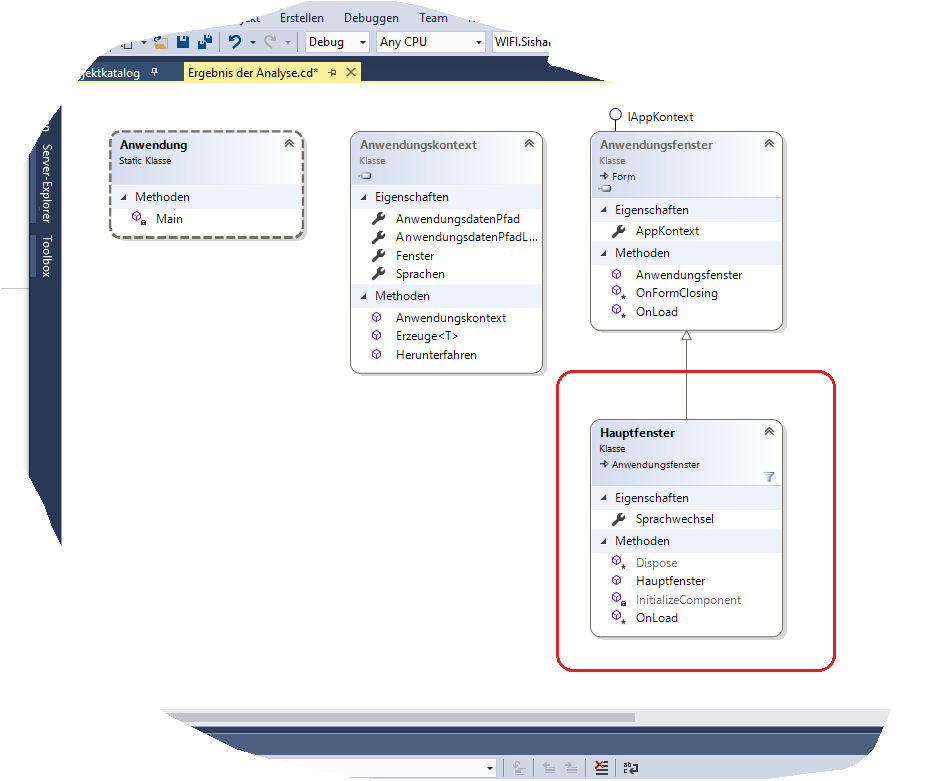
22. Jänner 2019

Zurück zur Oberfläche…



* Mehrere markierte Objekte können über das Menü Format ausgerichtet und verteilt werden
* Der Prototyp ist fertig

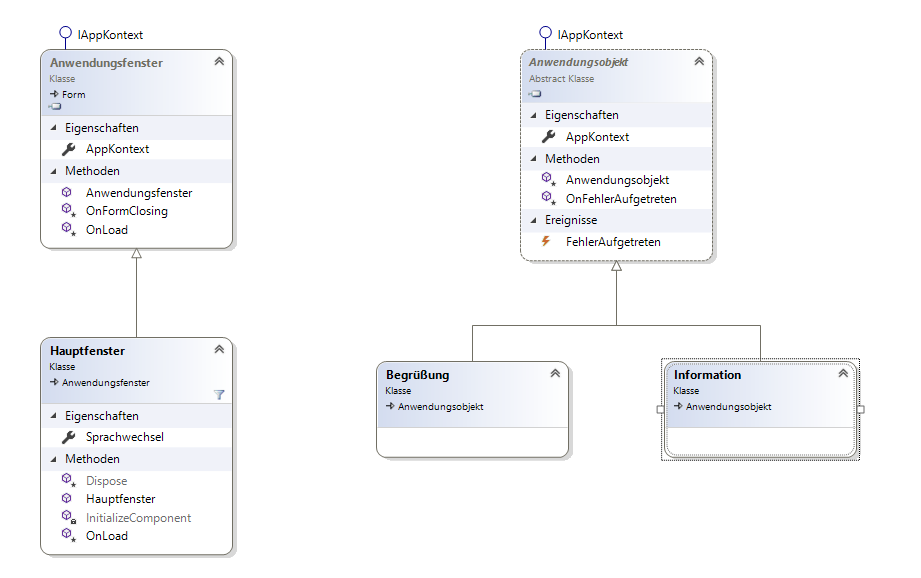
Wie geht’s weiter?



* Das Hauptfenster muss mehrere Darstellungen verwalten
* Die Begrüßung
* Eine Information

Das Hauptfenster kann nicht die Logik enthalten.   
Weil sonst verliert man komplett den Überblick.

* Wir benötigen zusätzliche (Anwendungs-)Objekte



* + Begrüßung und
  + Information

Dadurch ergibt sich ein neues Problem:

* Wer weiß, was da noch dazu kommt?
* Wir können nicht im Hauptfenster Variablen machen,   
    
  z. B.
  + Begrüßung AktuellesObjekt
  + Information AktuellesObjekt
  + … so wird das nix

Eine Möglichkeit

* + object AktuellesObjekt  
      
    Das ist late binding und wir müssen ständig…
  + Casten, Typbestätigungen durchführen
  + Uninteressant

Lösung:

* Eine Wunschliste des Hauptfensters an die zukünftigen Objekte
* Eine Interface

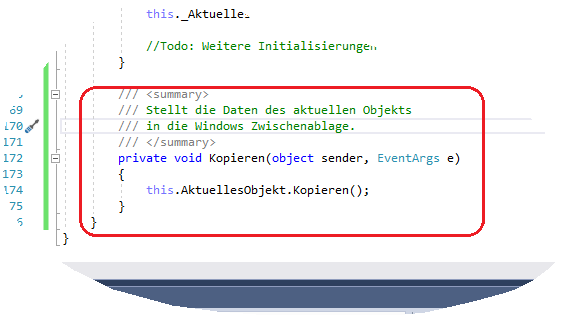
Warum nicht generisch?

* Weil die Begrüßung muss z. B. ein Bild von sich kopieren
* Die Information muss einen Text kopieren
* Das ist eine komplett andere Logik

Was muss sich das Hauptfenster wünschen?

* Grundsätzlich die Möglichkeiten zum …
  + Kopieren
  + Speichern
  + Druckansicht

Weil in der Wunschliste des Hauptfensters jetzt ein Kopieren vorhanden ist, kann der Befehl zum Kopieren im Hauptfenster fertiggemacht werden.



Die Abbildung zeigt den Ereignis Behandler für das Click Ereignis des AktionenKopieren Befehls.

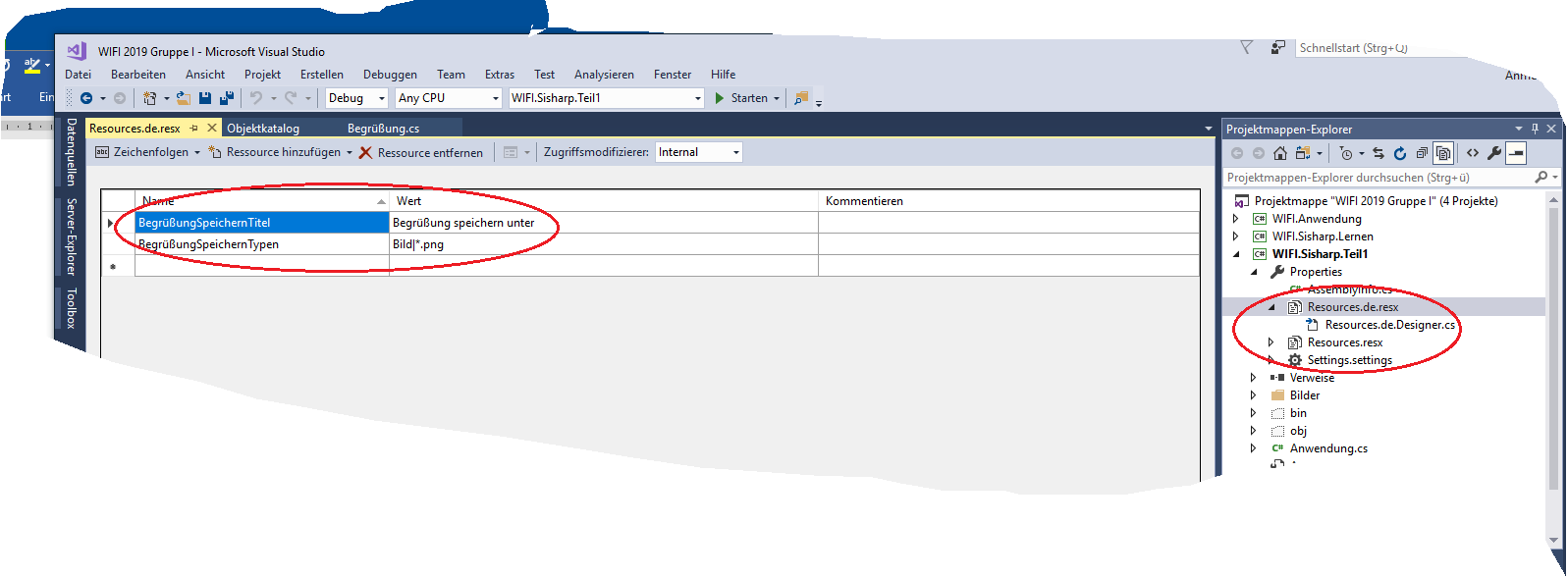
* Mehr bleibt im Hauptfenster nicht zu tun

Problem mit dem Lokalisieren in der Begrüßung / Information

* Im Hauptfenster wurde heute bereits lokalisiert, d.h. übersetzt
* Es wurde nur die Sprache geändert und die Texte der Steuerelemente übersetzt
* Die Begrüßung und die Information haben so etwas aber nicht

Was tun?

* Dasselbe wie im WIFI.Anwendung’s Assembly



d.h. in Projekteigenschaften die Ressourcen angegeben und die ganze Ressourcen-Datei kopieren, mit dem Sprachcode umbenennen und übersetzen.

24. Jänner 2019

Wie bringen wir dem PrintPreviewDialog bei,   
dass er sich die Fensterposition merkt?

* Grundsätzlich wird dazu
  + Unsere Infrastruktur benötigt, der AppKontext
  + Diesen würden wir über das „Anwendungsfenster“ bekommen
* Es gibt aber nur Einfachvererbung

Idee:

* Wir könnten im Hauptfenster, nach dem das Objekt PrintPreviewDialog erzeugt wurde, Ereignisbehandler an Load und FormClosing anhängen
  + Relativ schnell
  + Was machen wir, wenn der PrintPreviewDialog wieder benötigt wird? -> Wieder??? Nein
* Wie beim Formular, den PrintPreviewDialog erweitern
* Erweiterungsmethoden implementieren  
  (Wenn’s nicht wissen, was sonst tun sollen)

Zum Fragen beim Beenden…

* Auf keinen Fall bei einer Schaltfläche  
  oder einem Befehl
* Es gibt viele Möglichkeiten zum Beenden

Also wo?

* Beim Schließen – Ereignis

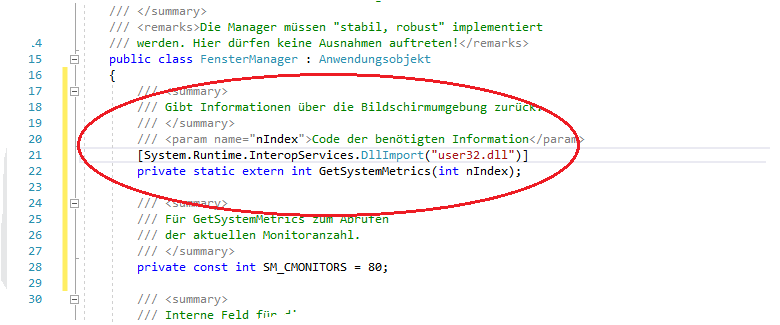
Beim Position Speichern der Seitenansicht festgestellt, dass die Anzahl der Bildschirme noch nicht berücksichtigt ist…

* Wir müssen unserem Fenstermanager das noch rasch beibringen

Wie kann man die Anzahl der Bildschirme, die Auflösung, … feststellen?

* C# -> hat nix
* .Net ?  
    
  Hat a nix

Dann müssen wir in’s Betriebssystem



* <https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/interop/>

Zum Zeichnen der Begrüßung

* Wir müssen die Begrüßung am Bildschirm zeichnen
* Wir müssen die Begrüßung als Bild in der Zwischenablage zeichnen
* Wir müssen die Begrüßung beim Speichern als Bild zeichnen
* Wir müssen die Begrüßung für das Ausdrucken zeichnen

Wenn wir das 4x machen, ist’s fad

Grundsätzlich:

* Windows Forms nutzt das alte Graphical Device Interface von Windows (GDI)
* Das GDI wurde in System.Drawing gekapselt
* Das Objekt zum Zeichnen: System.Drawing.Graphics
* Immer, wenn möglich, das Graphics Object vom Paint-Ereignis nehmen, damit das Bild nicht flackert

Praktisch:

* Der Bildschirm, die Bilder und der Ausdruck arbeiten alle mit demselben System.Drawing.Graphics

Nur ein Problem:

* Am Bildschirm, beim Kopieren und Speichern:  
    
  Keine Ränder
* Beim Drucken müssen wir Seitenränder beachten

Problem:

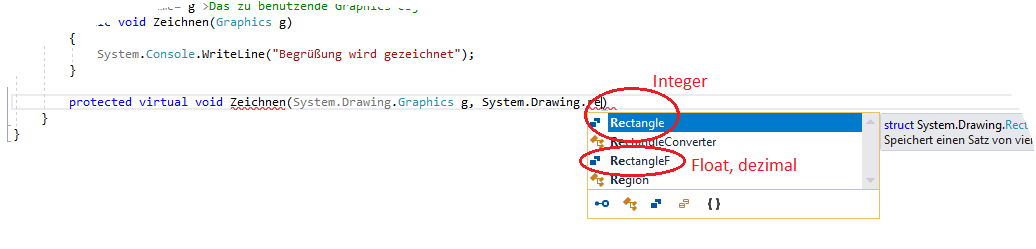
* Das IHauptfensterObjekt wünscht sich nur Zeichnen(Graphics g)
* Wir benötigten aber auch ein Zeichnen(Graphics g, Rectangle ränder)

Lösung:

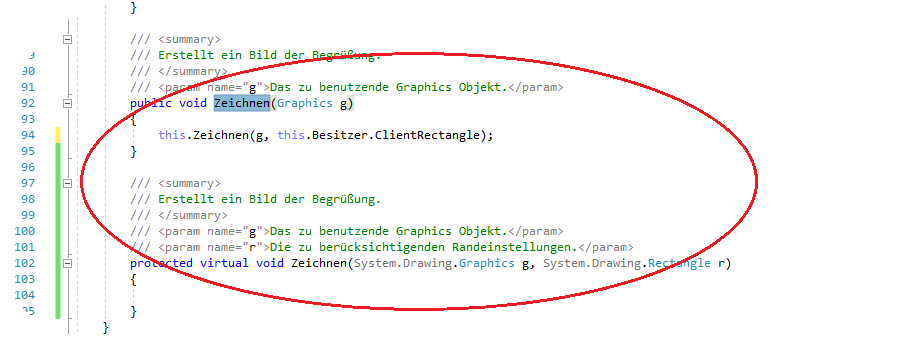
* Überladen

Hinweis:

Ständig sieht man Klassen zwei Mal…

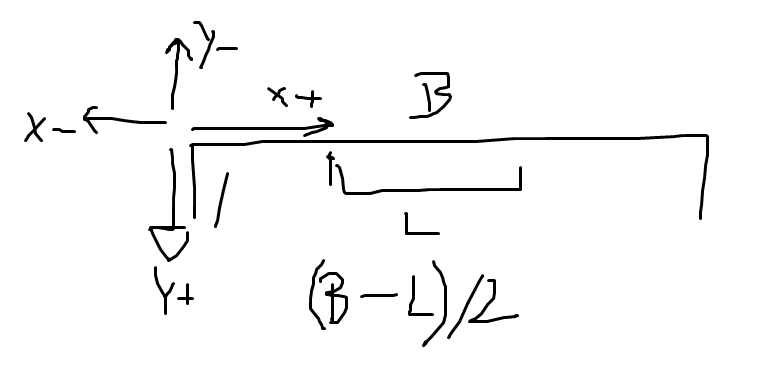


* Einmal „normal“: Ganzzahl Variante
* Einmal mit dem Suffix „F“: Die dezimal Variante (float)



Die Abbildung zeigt, wie die öffentliche Variante die interne   
mit den Rändern aufruft.

Zum Zentrieren eines Textes…



Nächster Punkt:

* Die Begrüßung.Kopieren() Methode implementieren

Es muss ein Bild sein, das in die Zwischenablage kommt.

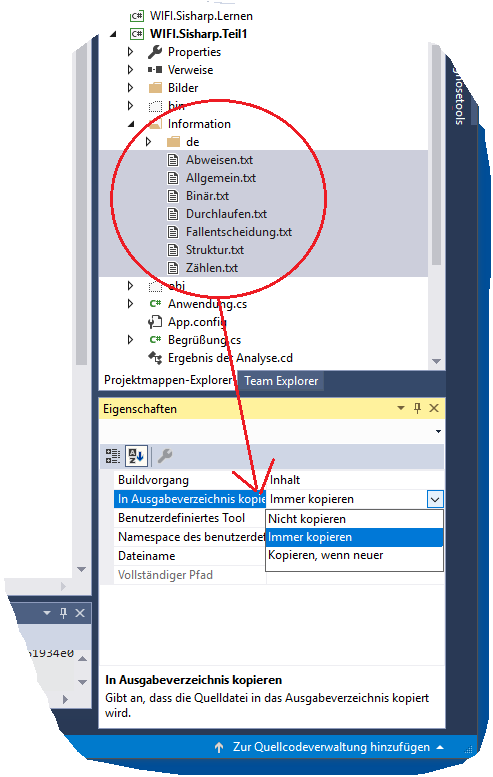
* Wo ist die Zwischenablage?

System.Windows.Forms.Clipboard

29.01.2019

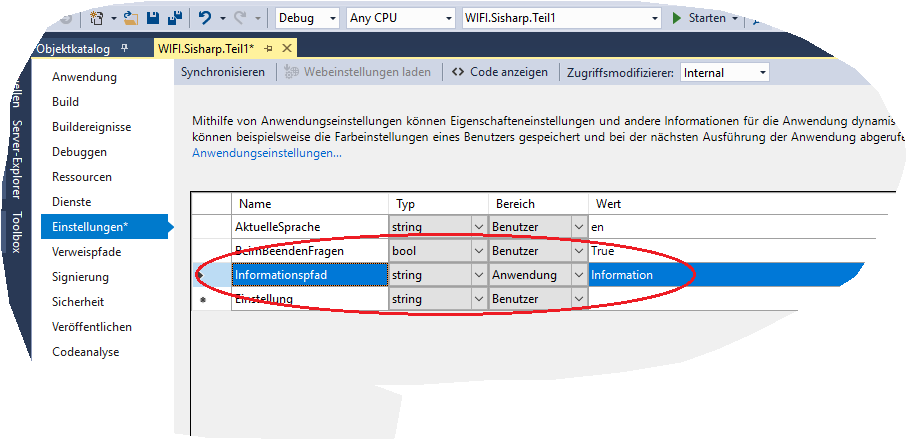
* Implementierung der WIFI.Anwendung.Verlauf Klasse
* Einbauen im OnLoad des Hauptfensters
* Internes Begrüßung.Zeichnen(…) fertiggemacht

Zu den anzuzeigenden Informationen:



* Wir wollen die Dateien nicht in die Ressourcen, damit der Inhalt einfacher „überarbeitet“ werden kann
* Deshalb werden die Textdateien beim Erstellen immer in einem eigenen Ordner „Information“ in das Verzeichnis der ausführbaren Datei kopiert

Damit die Anwendung …

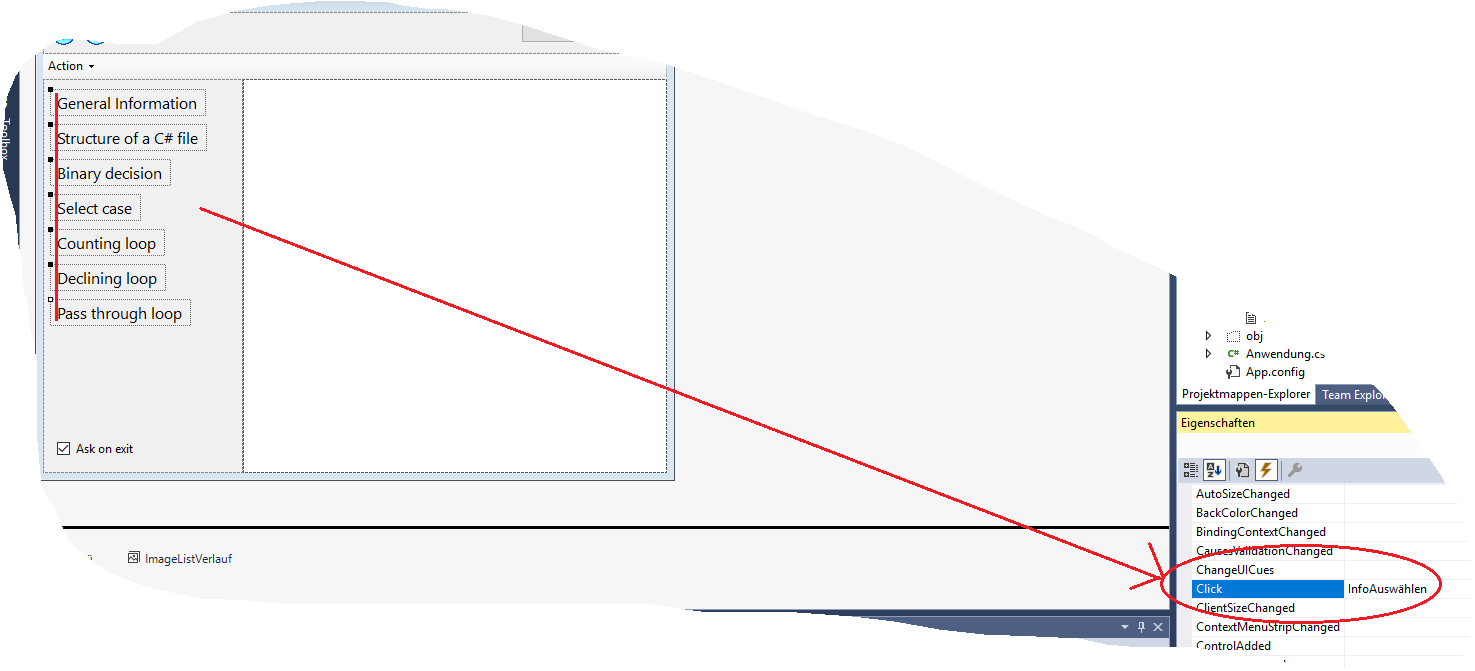


… den Pfad der Informationsdateien kennt, wird eine neue Anwendungskonfiguration hinzugefügt.

Unterschied zwischen „Benutzer“ und „Anwendung“ Einstellungen?

* Das Studio programmiert im Hintergrund
* „Benutzer“ Einstellungen werden eine Eigenschaft mit get und set
* „Anwendung“ Einstellungen werden Eigenschaften nur mit get. Also schreibgeschützt und können nur von den Administratoren geändert werden

Hinweis:



* Derselbe Ereignis Behandler kann unterschiedliche Ereignisse und Objekte benutzt werden, solange die Signatur,   
  d.h. der zweite Parameter e denselben Typ hat.

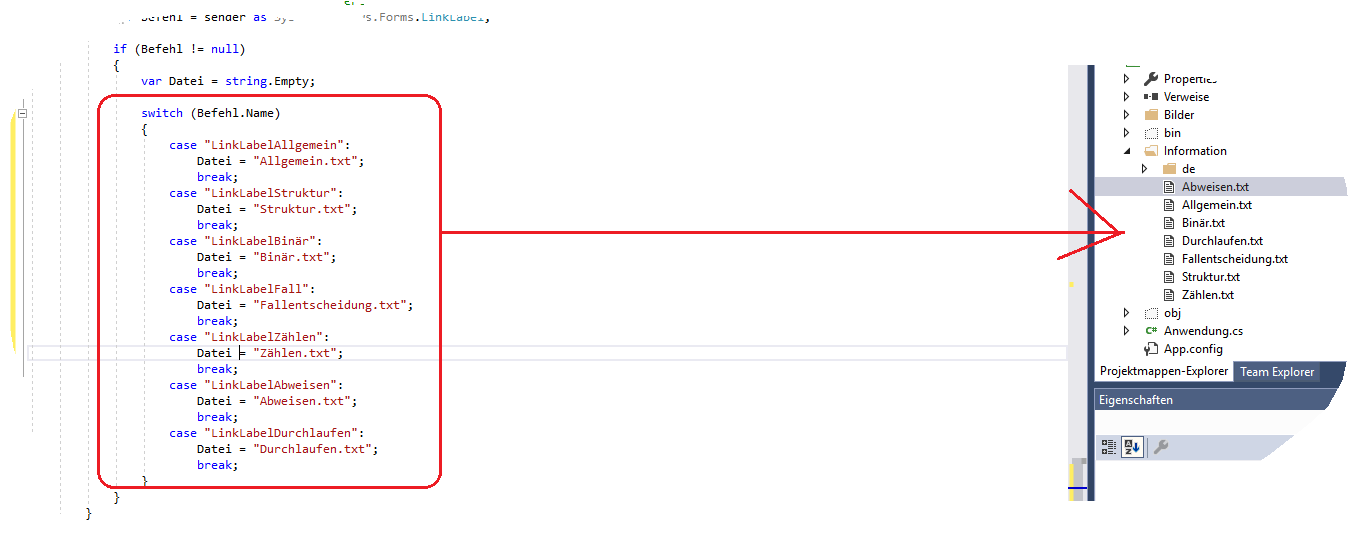
31.1.2019

Vorausschau, worum’s im Teil 2 geht:

* WPF (Windows Presentation Foundation)
* Entwurfsmuster Modell (Daten + Manager) View (Oberfläche) ViewModel (Manager zum Verwalten der Oberfläche)
* MVVM lose gekoppelt
* Im Teil 1 wurde im Formular der „Code Behind“ benutzt
* Client / Server Anwendung,   
  d.h. die „Datenlogik“, die Assembly, die mit der Datenbank kommuniziert (SQL Server Datenbank mit gespeicherten Prozeduren) wird hinter einer Web-Anwendung „versteckt“ und über einen SOAP Webdienst bereitgestellt

-

Die unterschiedlichen Dateien werden am …

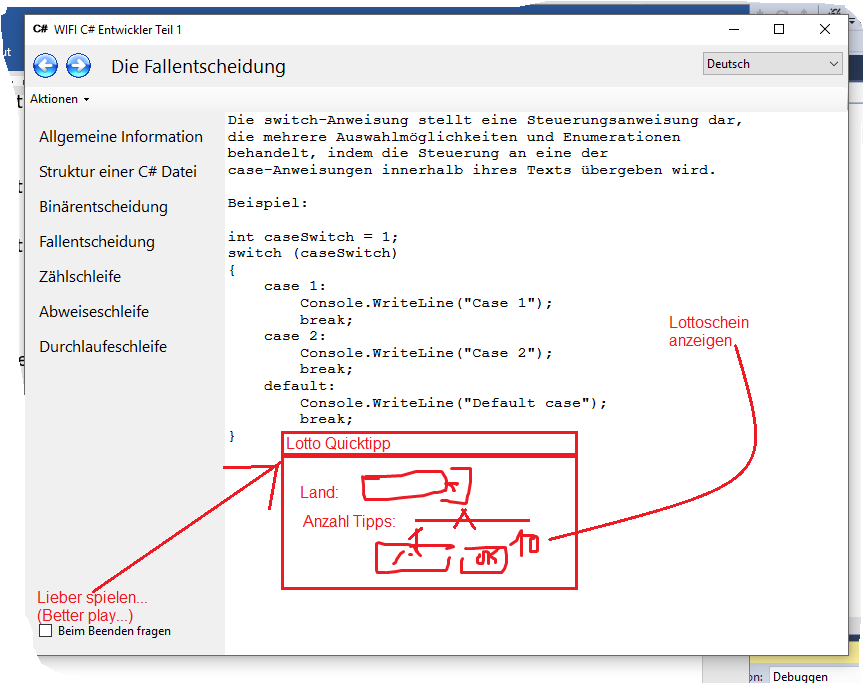


… Besten mit einer Fallentscheidung ausgewählt.

* Implementierung der WIFI.Sisharp.Teil1.Information – Klasse
* Implementierung von InfoAuswählen im Hauptfenster
* Implementierung von ImVerlaufBewegen im Hauptfenster

Wie könnte es weitergehen?

* Der Auftraggeber sagt, die Anwendung wird zu wenig benutzt
* Idee:



(Die Länder und das Tippberechnen aus dem WIFI.Lernen beziehen

Für die Anzeige des Scheins, einfach die Information-Klasse kopieren und auf „Lotto“ ändern