

Modul 318

Visual Studio





Autor: Urs Nussbaumer

Version: 1.1

Datum: 12.04.2017



Inhaltsverzeichnis

Inhaltsverzeichnis	2
1 Visual Studio Entwicklungsumgebung (IDE)	3
1.1 Erster Start und Konfiguration	3
1.1.1 Ablageort für Projekte und Entwicklungseinstellungen festlegen	3
1.1.2 Hilfe / Dokumentation konfigurieren (Visual Studio 2015).....	3
1.2 Die Bestandteile der Entwicklungsumgebung	4
1.2.1 Übersicht	4
1.2.2 Die Symbolleiste	4
1.3 Neues Projekt erstellen (Windows-Anwendung)	5
1.3.1 Formular umbenennen	6
1.3.2 Formular hinzufügen.....	6
1.4 Der Projektmappen-Explorer	7
1.5 Das Design-Fenster	8
1.6 Das Codefenster	8
1.6.1 Intelli Sense	9
1.7 Eigenschaften Fenster	10
1.8 Programm starten/stoppen	10
1.9 Toolbox	11
1.9.1 Die wichtigsten Steuerelemente	11
1.9.2 Vorgefertigte Dialogfelder (in der Toolbox ganz unten)	18
1.9.3 Zusätzliche Komponenten und Steuerelemente	18
1.10 Ressourcen hinzufügen.....	19

1 Visual Studio Entwicklungsumgebung (IDE)

1.1 Erster Start und Konfiguration

Wählen Sie beim ersten Start die Einstellung **Allgemeine Entwicklungseinstellungen**.

Tipp: Falls Sie eine andere Auswahl getroffen haben: Menü *Tools* → *Import and Export Settings* → *Reset all Settings*

1.1.1 Ablageort für Projekte und Entwicklungseinstellungen festlegen

Arbeiten Sie lokal. Erstellen Sie z.B. einen Ordner "C:\Modul_318\".

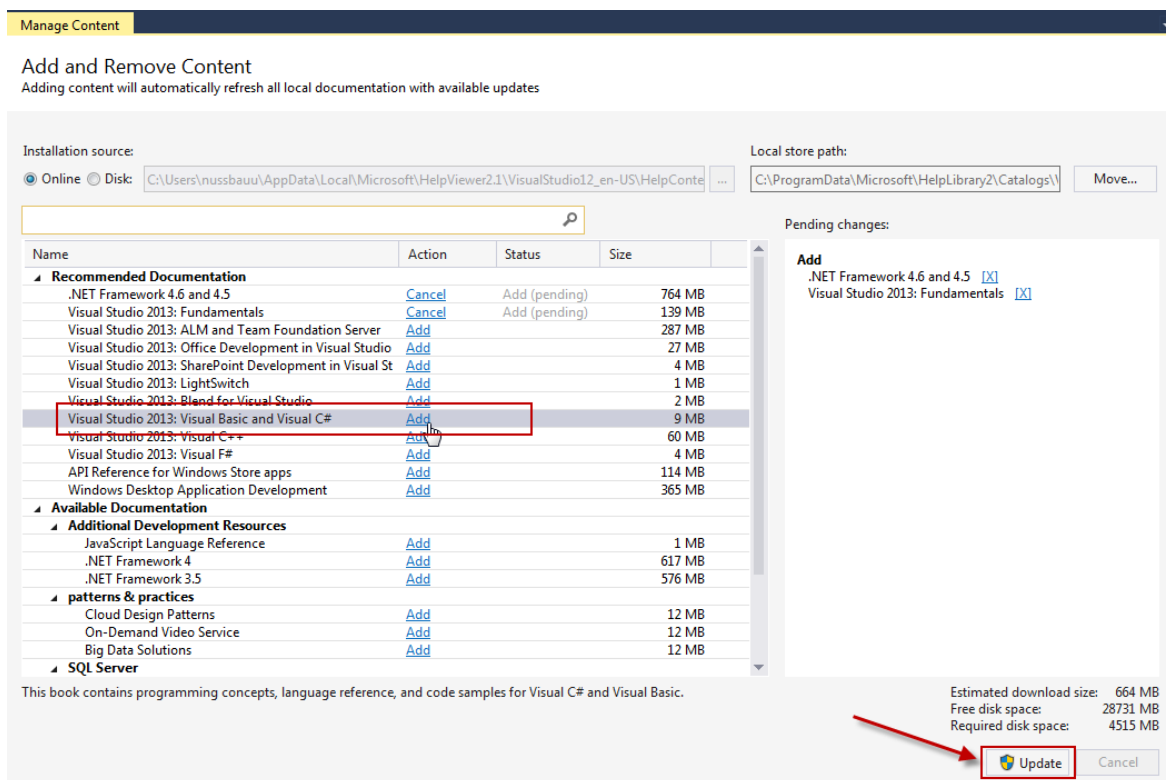
Setzen Sie in Visual Studio, Menü *Tools* → *Options...* → *Projects and Solutions* die „Projects location“ auf diesen Ordner.

Tipps:

- Kopieren Sie täglich ihren gesamten Projekt-Ordner auf ein persönliches externes Laufwerk, z.B. einen USB-Stick (Datensicherung).
- Erstellen Sie im Projektordner für jede Übungsaufgabe ein neues Projekt!

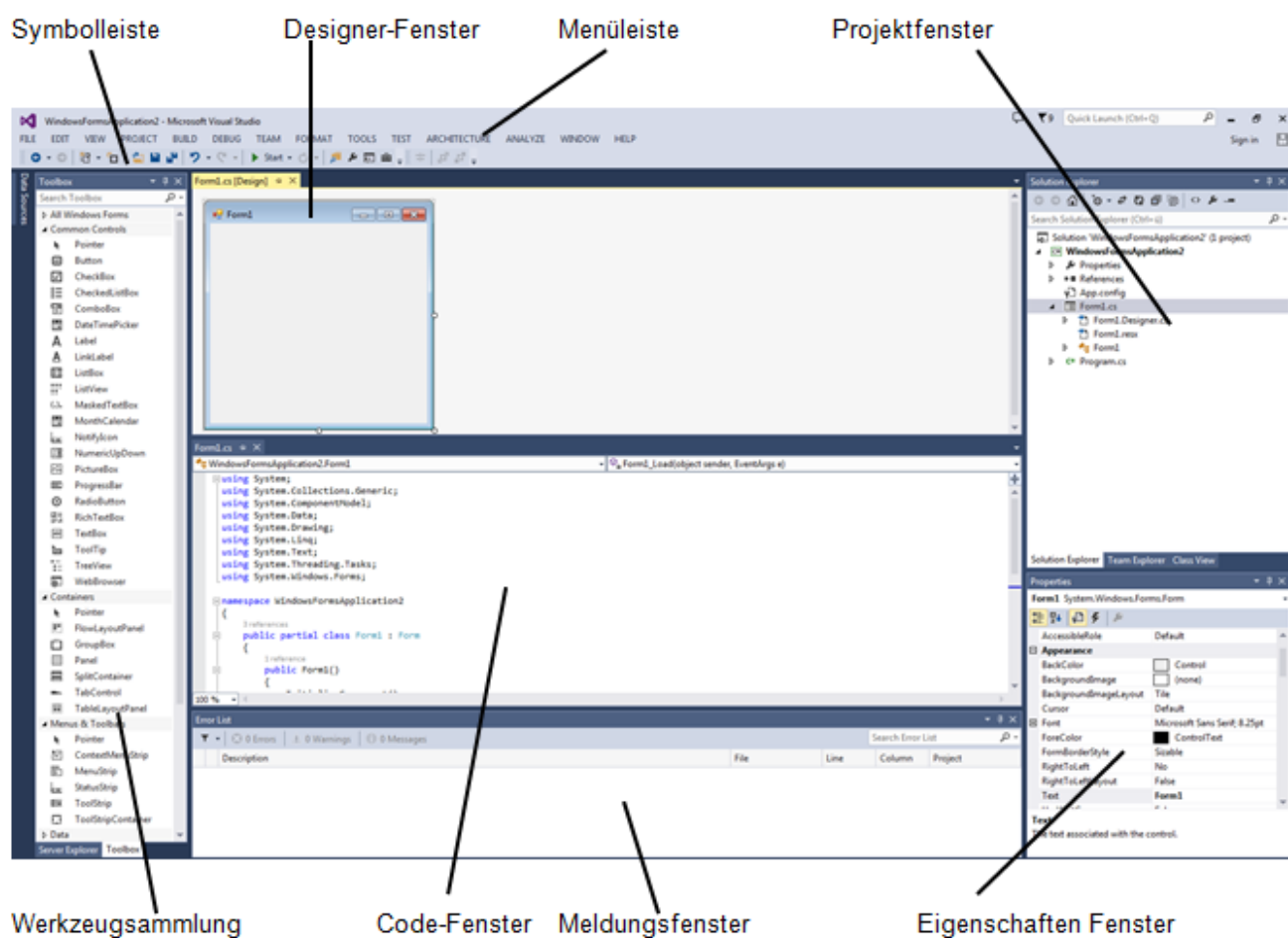
1.1.2 Hilfe / Dokumentation konfigurieren (Visual Studio 2015)

Menu *Help* → *Add and Remove Help Content* startet den Microsoft Help Viewer über den Hilfspakete heruntergeladen werden können. Fügen Sie Hilfe-Dokumentationen für das .NET Framework und für Visual C# hinzu (falls nicht schon hinzugefügt...) und klicken Sie auf „Update“.



1.2 Die Bestandteile der Entwicklungsumgebung

1.2.1 Übersicht



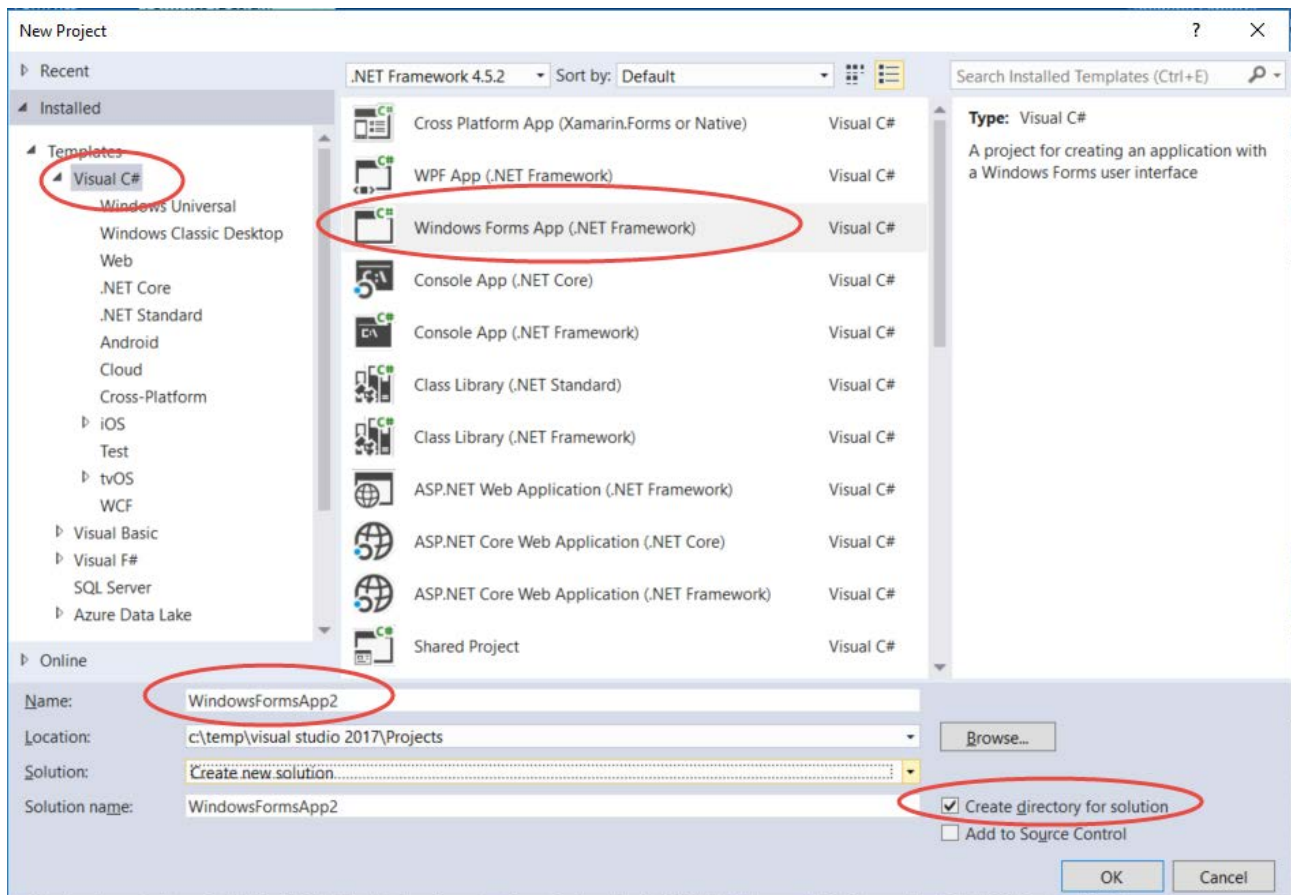
1.2.2 Die Symboleiste

Neues Projekt Alles speichern Programm starten (F5)



1.3 Neues Projekt erstellen (Windows-Anwendung)

Rufen Sie das Menu → *File* > *New* > *Project...* auf. Es öffnet sich das folgende Dialogfenster.



- *Windows Forms App* → Erstellt eine ausführbare Windows Dialog-Anwendung
- *Console Application* → Erstellt eine ausführbare Anwendung ohne Oberfläche

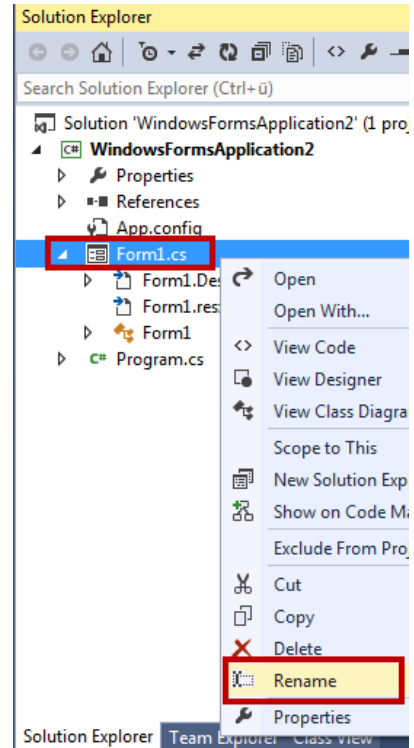
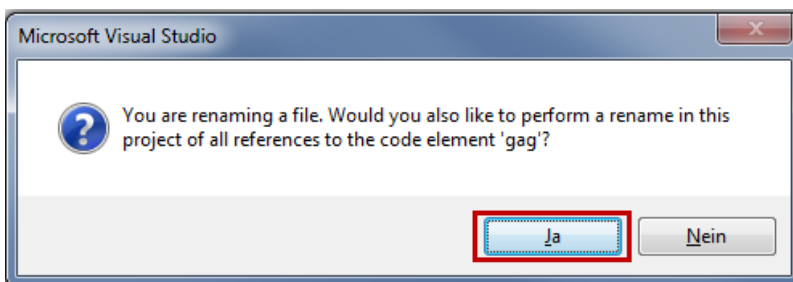
Wichtig: Geben Sie den Namen Ihres Projektes **VOR** dem Speichern (also bevor Sie OK klicken) an!

1.3.1 Formular umbenennen

Forms werden auch als Dialoge oder Formulare bezeichnet. Sie stellen die Benutzeroberfläche (GUI) des Programms dar. Auf einer Form befinden sich TextBoxen, Listen, Buttons, Label, Bilder usw.

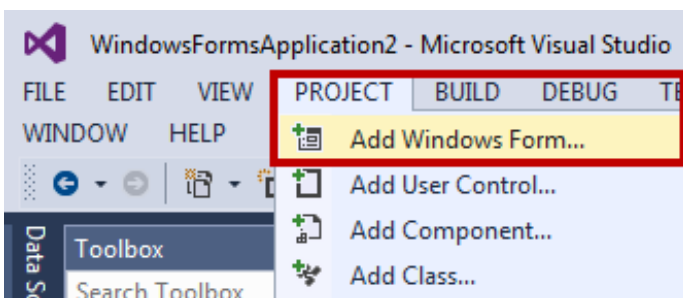
Nach Anlegen eines neuen Projekts befindet sich standardmässig bereits ein Formular mit Namen „Form1“ in dem Projekt. Sie finden es im Projektmappen-Explorer (Solution Explorer). Diesem Formular können Sie nun einen sinnvolleren Namen geben, indem Sie es im Projektmappen-Explorer auswählen und dann über das Kontext-Menu der rechten Maustaste umbenennen.

Wichtig: Die auf die Umbenennung folgende Frage unbedingt mit „Ja“ quittieren!

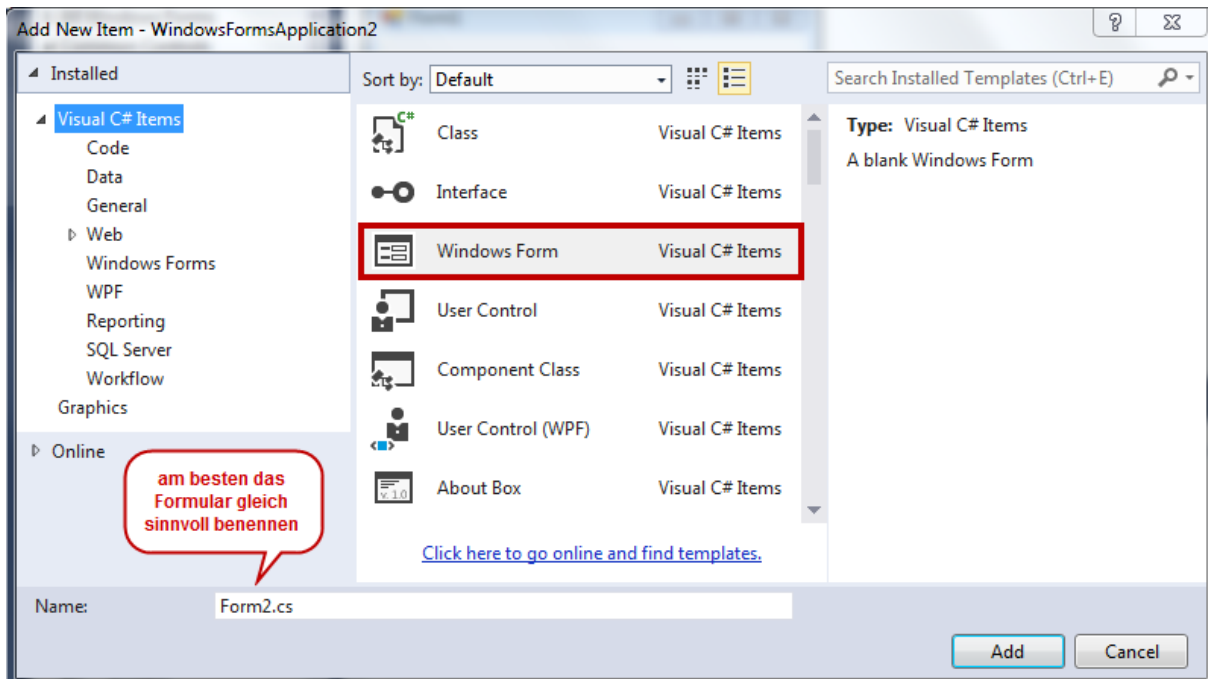


1.3.2 Formular hinzufügen

Der Dialog "Add New Item" (Menu *Project > Add Windows Form...*) dient dazu, neue respektive weitere Forms in ein bestehendes Projekt einzufügen. 8-ung: Wählen Sie bei den installierten Vorlagen „Windows Forms“.



Wählen Sie im Dialog „Add New Item“ nun aus der Liste die Vorlage „Windows Forms“ aus und geben Sie dem Formular am besten gleich sofort einen sinnvollen Namen.



Tipp: Formular sofort benennen!

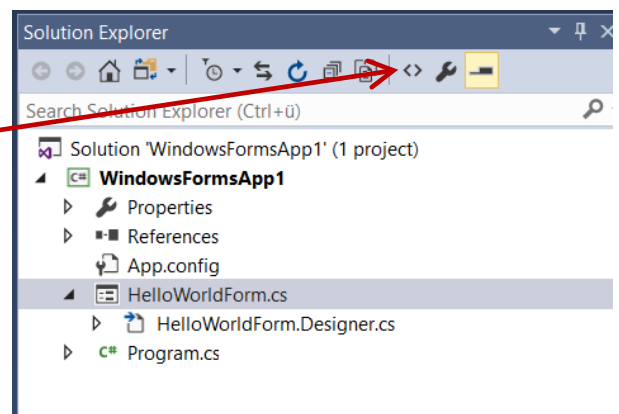
Tipp: Bestimmen Sie, welches Formular beim Programmstart angezeigt werden soll, indem Sie die Datei Program.cs ändern: `Application.Run(new MyForm());`

1.4 Der Projektmappen-Explorer

Der Projektmappen-Explorer (engl.: Solution Explorer) zeigt eine Liste an mit den Projekten und allen Elemente, die in einem Projekt enthalten sind.

Code anzeigen:

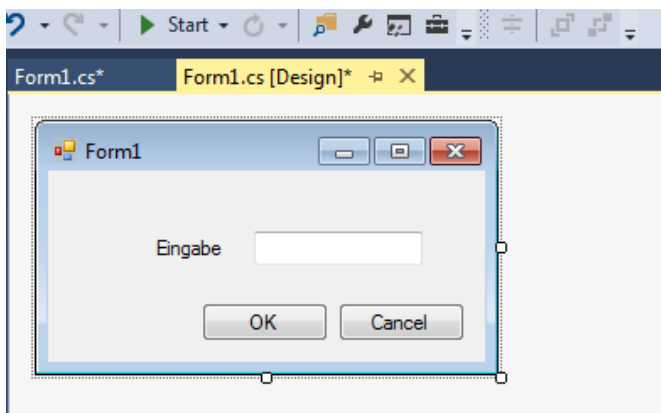
Zeigt das Codefenster des ausgewählten Elementes an um den Code zu erstellen und zu bearbeiten.



1.5 Das Design-Fenster

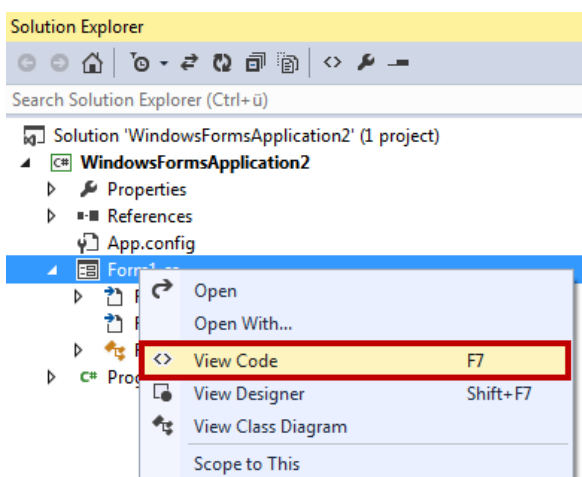
Im Designer können Sie ein Formular mit Steuerelementen, also zum Beispiel mit Buttons oder Text-boxen, visuell entwerfen und gestalten. Die Steuerelemente gehören zum Formular und werden zur Laufzeit zusammen mit diesem erzeugt und angezeigt.

Sie wechseln vom Codefenster (siehe unten) zum Designfenster, indem Sie die betr. Form/Klasse im Projektmappen-Explorer auswählen und über das Kontextmenu der rechten Maustaste den Eintrag *Designer anzeigen* (engl.: *View Designer*) wählen.

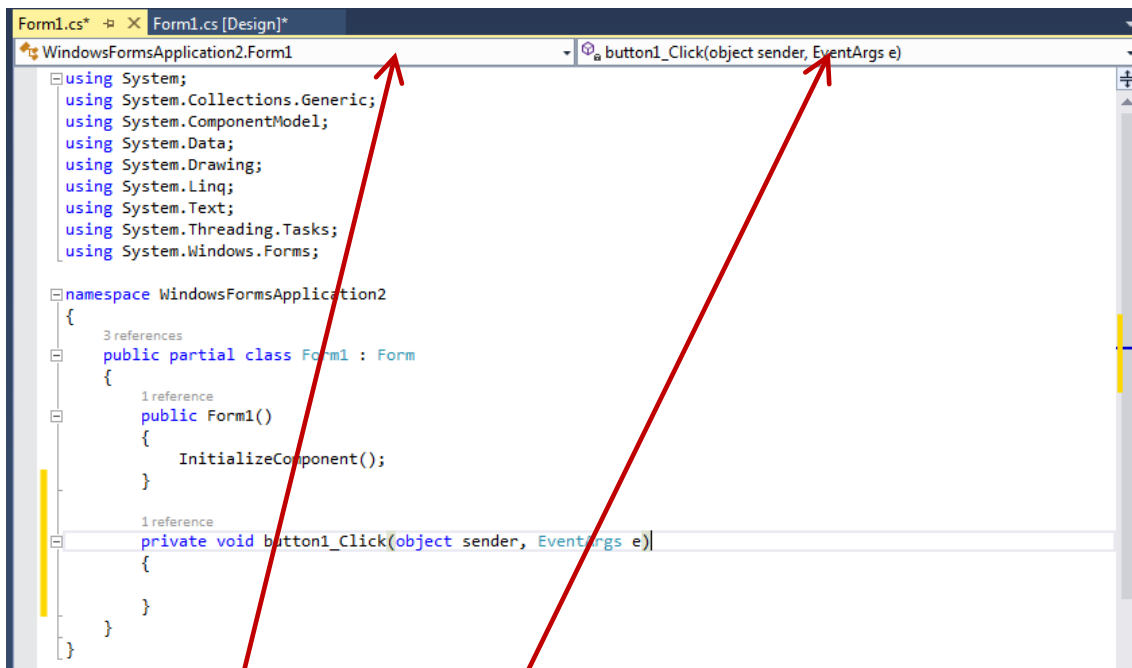


1.6 Das Codefenster

Sie öffnen das Codefenster, indem Sie die betr. Form/Klasse im Projektmappen-Explorer auswählen und über das Kontextmenu der rechten Maustaste den Eintrag *Code anzeigen* (engl.: *View Code*) wählen.



Im Codefenster wird C# - Code erstellt und bearbeitet. Es können so viele Codefenster geöffnet werden, wie Klassen vorhanden sind, so dass der Code auf einfache Weise in den verschiedenen Klassen angezeigt und zwischen diesen kopiert und eingefügt werden kann.



Dropdownfeld Objekt

Zeigt den Namen des markierten Objekts an. Ein Klick auf den Dropdown-Pfeil zeigt eine Liste der verschiedenen Objekte die mit dem Formular verknüpft sind.

Dropdownfeld Methode / Ereignis

Listet alle Methoden oder Ereignisse auf, die von C# für das im Objekt-Feld ausgewählte Formular oder Steuerelement verknüpft wurden. Zusätzliche Ereignisse können über das Eigenschaftsfenster des Objekts verknüpft werden.

1.6.1 Intelli Sense

Im Codefenster werden Sie unterstützt durch die Funktion „IntelliSense“ von Visual Studio beim Schreiben von Code. Immer, wenn Sie Code eintippen, erscheint eine Liste von möglichen Ergänzungen:

- Auswahl der Vorschläge ändern: {Up} / {Down}
- Auswahl übernehmen: {Tab}
- Auswahl übernehmen und einen Leerschlag einfügen: {Space}

1.7 Eigenschaften Fenster

Ein wichtiges Fenster ist das Eigenschaften-Fenster (engl.: Properties Window). Es zeigt für ein im Designer ausgewähltes Steuerelement dessen aktuellen Zustand (Entwurfszeiteigenschaften) an. Über das Eigenschaften-Fenster können die Eigenschaften der Steuerelemente zur Entwurfszeit geändert werden.

Feld "Objekt"

Zeigt das aktuell im Formular markierte Element an.

Toolbarbuttons "Nach Kategorien" / "Alphabetisch"

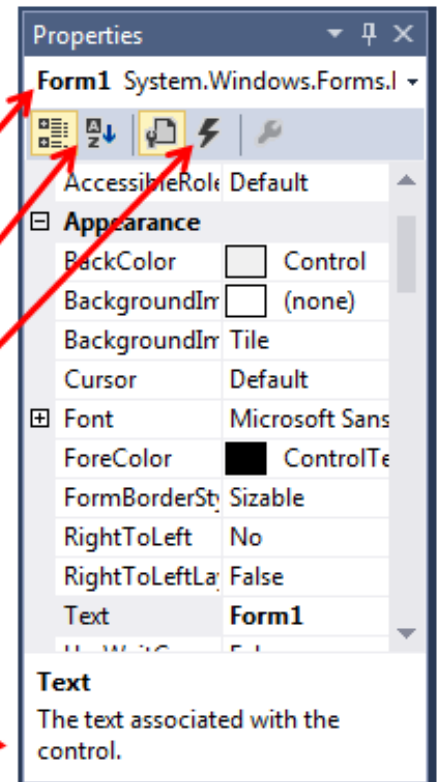
Listet die Eigenschaften oder Ereignisse des Elements nach Kategorien gruppiert oder alphabetisch sortiert auf.

Toolbarbuttons "Eigenschaften" / "Ereignisse"

Listet die Eigenschaften des Elements oder seine Ereignisse auf.

Beschreibungsbereich

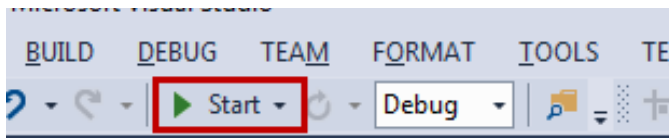
Kurzbeschreibung der ausgewählten Eigenschaft oder des Ereignisses



1.8 Programm starten/stoppen

Kompilieren und Starten/Debuggen → Taste <F5>

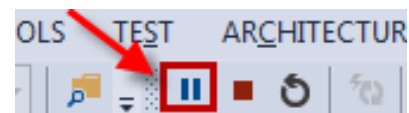
Die Schaltfläche *Start* führt das aktuell geöffnete Projekt aus (als wäre es eine EXE-Datei).



Tipp: Bei einer Konsolen-Anwendung wird das cmd-Fenster am Ende sofort geschlossen, und die letzten Ausgaben sind somit nicht sichtbar. <CTRL> + F5 lässt das Konsolenfenster bei Anwendungsende noch offen (Debugging ist dann aber nicht mehr möglich).

Unterbrechen → Tasten <CTRL> + <BREAK>

Die Schaltfläche *Unterbrechen* unterbricht die momentane Ausführung eines Projektes (funktioniert auch bei Endlosschleifen). Weiterfahren mit F5 oder im Einzelschritt-Modus.



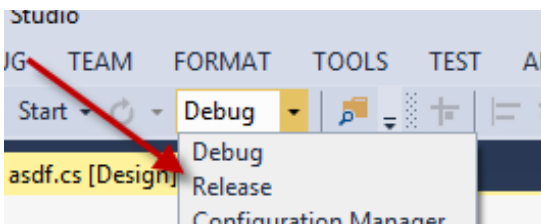
Beenden

Diese Schaltfläche beendet das momentan laufende Programm (auch nach „Unterbrechen“ nötig).



Release-Datei erstellen

Die Debug-Konfiguration verwenden wir für die Programmentwicklung, die Release-Konfiguration für die Programmauslieferung. Eine Release-Konfiguration enthält keine Informationen zur Fehlersuche hat optimierten Code. Auch braucht sie keine speziellen Debug-Bibliotheken.



1.9 Toolbox

Nachfolgend sind einige Beispiele von Steuerelementen mit häufig benutzten Eigenschaften, Ereignissen und Methoden aufgeführt.

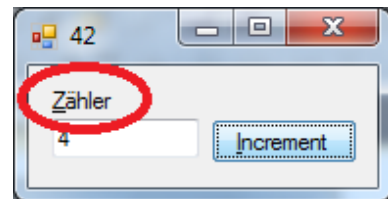
1.9.1 Die wichtigsten Steuerelemente

Label

Zweck

Das Label ist für das Anzeigen von fixem Text gedacht. Der angezeigte Text kann vom Benutzer weder verändert noch selektiert werden.

Falls ein Label einen Tastatur-Shortcut besitzt (unterstrichenes Zeichen, sichtbar ev. erst nach Drücken von <ALT>) wird damit das darauffolgende Element gewählt.



Eigenschaften

(Name)	lbl...
Text	Beschriftung des Labels. Ein & vor einem Zeichen markiert einen Tastatur-Shortcut (im Beispiel: "&Zähler" → mit <ALT>+Z wird das nächste Feld gewählt, hier das Textfeld mit dem Inhalt "4").
AutoSize	True bewirkt, dass sich die Länge des Labels der Textlänge anpasst. ¹

Tipp

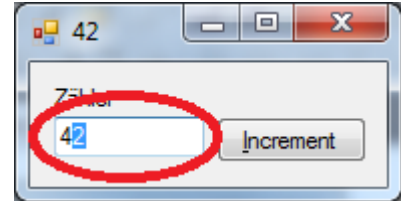
Verwenden Sie keine Labels für die Anzeige von Resultaten oder wichtigen Informationen. Der Benutzer kann den Text sonst nicht auswählen und kopieren. Für eine reine Datenausgabe verwenden Sie bevorzugt eine TextBox bei der Sie ReadOnly auf True setzen.

¹ Sind AutoSize und WordWrap beide False und ist der Text länger als das Label, wird der Rest des Texts einfach abgeschnitten (nicht angezeigt).

TextBox

Zweck: Die TextBox ist das grundlegende Ein- und Ausgabesteuerelement, in dem der Benutzer Text eingeben, auswählen und bearbeiten kann.

Der Text kann natürlich auch durch den Code im Programm bestimmt werden.

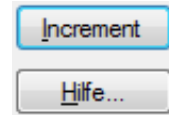


Eigenschaften	(Name)	txt...
	Text	Angezeigter Text in der Textbox (vom Benutzer eingegeben oder aus Ihrem Code festgelegt).
	Multiline	True erlaubt automatischen Umbruch der Ein- oder Ausgabe auf mehrere Zeilen.
	ScrollBars	Bestimmt, ob es in der TextBox keine, eine horizontale, vertikale oder beide Bildlaufleisten hat.
	AcceptsReturn	Gibt für eine Multiline-TextBox an, ob Zeilenumbrüche (Return, Enter) eingegeben werden können oder ob damit die Default-Aktion der Form ausgelöst werden soll.
	ReadOnly	True erlaubt keine Eingaben ins Feld (der Inhalt der TextBox kann vom Benutzer aber noch immer selektiert / kopiert werden)
Ereignisse	TextChanged	Wird ausgelöst sobald sich der Inhalt der TextBox ändert. Dies geschieht bei Eingabe von „Hallo“ fünf mal.
	GotFocus	Wird ausgelöst wenn die TextBox angewählt wird (mit Maus oder Tastatur oder bei Aktivierung der Anwendung).
	LostFocus	Wird ausgelöst wenn die TextBox verlassen wird (Ende der Eingabe oder Wechsel der Anwendung).

- Tipps**
1. Für eine Resultat-Anzeige ist eine ReadOnly TextBox ist meist sinnvoller als ein Label, damit der Benutzer den Inhalt kopieren kann.
 2. Bei einer ReadOnly TextBox wird meist TabStop auf False gesetzt. So wird der normale <TAB>-Ablauf nicht gestört. Geben Sie dem die TextBox beschreibenden Label aber dennoch einen Tastatur-Shortcut!

Button

Zweck: Buttons eignen sich dafür, vom Benutzer auf Mausklick oder Tastendruck Aktionen ausführen zu lassen.



Eigenschaften:

(Name)	btn....
Text	Diese Eigenschaft legt die Beschriftung fest.

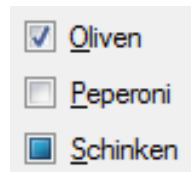
Ereignisse: Click Wird ausgelöst, wenn der Benutzer den Button drückt.

Verwandte:

Form: AcceptButton	Pro Form kann ein Button als AcceptButton werden. Dieser Button wird hervorgehoben dargestellt, und ein <RETURN> in dieser Form löst ein Click-Ereignis für den markierte Button aus (wenn der Fokus nicht gerade in einer TextBox mit AcceptsReturn = True ist). Diese Eigenschaft wird meist auf eine OK Schaltfläche gesetzt.
Form: CancelButton	Pro Form kann ein Button als CancelButton angegeben werden. Ein <Esc> in dieser Form löst ein Click-Ereignis für den markierten Button aus. Diese Eigenschaft wird meist auf eine Abbrechen- / Cancel-Schaltfläche gesetzt.

CheckBox

Zweck: Die CheckBox ist ein Steuerelement, mit welchem Sie eine Option auswählen (Anzeige mit Häkchen) oder nicht (Kästchen leer). Sie kann manchmal noch einen dritten Zustand annehmen: grau → unbestimmt oder widersprüchlich.



Eigenschaften:

(Name)	chk....
Text	Beschriftung der CheckBox.
ThreeState	False wenn Benutzer die Box nur an- oder abwählen kann.
Checked	True wenn die Box ausgewählt wurde, False wenn die Box nicht ausgewählt wurde (ThreeState: → CheckState).
CheckState	Zustand einer einfachen oder ThreeState-CheckBox: Checked, Unchecked oder Indeterminate.

Ereignisse:

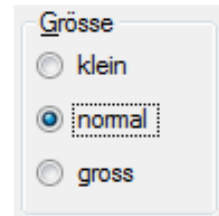
CheckedChanged	Der Zustand der CheckBox hat sich geändert.
CheckedStateChanged	Der Zustand einer einfachen oder ThreeState-CheckBox hat sich geändert.

Tipp Eine im Designer angewählte CheckBox sendet zur Laufzeit während der Dialog-Initialisierung bereits die Ereignisse *CheckedChanged* und *CheckedStateChanged*.

RadioButton

Zweck: RadioButtons erlauben es, aus mehreren Möglichkeiten eine auszusuchen. RadioButtons treten immer in Gruppen auf. Pro Gruppe kann nur ein RadioButton aktiviert sein.

Alle RadioButtons einer Gruppe müssen in einem Container-Element definiert sein (z.B. Form, GroupBox oder Panel). Bei mehreren Optionsgruppen brauchen Sie mehrere Container-objekte in denen Sie die RadioButtons platzieren (siehe "GroupBox").



Eigenschaften:

(Name)	opt....
Text	Beschriftung des RadioButton.
Checked	True wenn die Option ausgewählt wurde, False wenn die Option nicht ausgewählt wurde.

Ereignisse:

CheckedChanged	Der Zustand des RadioButtons hat sich geändert.
----------------	---

Tipps

1. Bei RadioButtons sollte die TabStop Eigenschaft auf False gesetzt sein.
2. Definieren Sie keine Shortcuts für die einzelnen RadioButtons sondern einen Shortcut für das Container-Element.

GroupBox

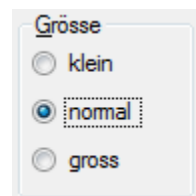
Zweck:

Benutzersicht:

- Optische Gruppierung von Elementen des Benutzerinterfaces.

Programmierersicht:

- Fasst mehrere RadioButtons zu einer Gruppe zusammen.
- Elemente als Gesamtes aktivieren / deaktivieren (Eigenschaft Enable) und anzeigen / verstecken (Eigenschaft Visible).

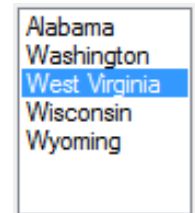


Eigenschaften:

(Name)	grp....
Text	Beschriftung des RadioButton.
Enable	False bewirkt, dass alle in der GroupBox enthaltenen Elemente ebenfalls nicht wählbar sind.
Visible	False bewirkt, dass alle in der GroupBox enthaltenen Elemente ebenfalls versteckt / unsichtbar werden.

ListBox

Zweck: Die ListBox bietet dem Benutzer die Möglichkeit, aus einer ein- oder mehrspaltigen Liste ein oder mehrere Elemente auszuwählen.

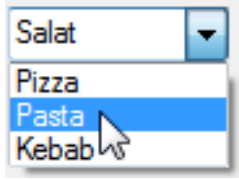


Eigenschaften:	(Name)	Ist....
	SelectionMode	One Max. ein Eintrag wählbar Multi.... Mehrere Einträge wählbar
	Sorted	True Eintrag werden alphabetisch sortiert.
	Items	Gibt den Inhalt der List zurück. Items ist eine ObjectCollection-Objekt (→ <i>Methoden</i>).
	SelectedIndex	Gibt den (0-basierten) Index des gewählten Elements zurück oder legt ihn fest (z.B. beim Initialisieren der ListBox). Gibt -1 zurück falls kein Eintrag gewählt ist. Verwenden Sie SelectedIndices für eine Multiselect-ListBox.
	SelectedItem	Gibt das gewählte Element zurück oder legt es fest (z.B. beim Initialisieren der ListBox). Gibt Nothing zurück falls kein Eintrag gewählt ist. Verwenden Sie SelectedItems für eine Multiselect-ListBox.
Methoden:	Items.Add()	Fügt der Auswahlliste ein Element hinzu. Bsp: <code>cboFood.Items.Add("Pizza")</code> <code>cboFood.Items.Add("Pasta")</code> <code>cboFood.Items.Add("Kebab")</code>
	Items.Remove()	Entfernt das angegebene Element aus der Auswahlliste. Ähnlich ist Items.RemoveAt().
	Items.Clear()	Entfernt alle Einträge der Liste.
	Items.Contains()	Prüft, ob ein Element in der Dropdownlist vorhanden ist. Ähnlich ist Items.IndexOf().
Ereignisse:	SelectedIndexChanged	Die Elemente-Auswahl hat sich geändert.
	DoubleClick	Doppelklick auf ein Element (→ lösen Sie die Default-Aktion dazu aus).

Tipps

1. Für mehrspaltige Einträge in Listboxen ist die Eigenschaft MultiColumn **nicht** geeignet. Verwenden Sie stattdessen ein ListView-Objekt mit View = Details.
2. Meist empfiehlt es sich, eine Start-Selektion festzulegen. Beispiel:
`lstStates.SelectedIndex = lstStates.Items.Add("New York")`

ComboBox

Zweck:	Die ComboBox bietet dem Benutzer die Möglichkeit, aus einer Liste ein Element auszuwählen (DropDownList) oder auch einen neuen Wert angeben (DropDown).	
		
Eigenschaften:	(Name)	cbo....
	Text	Gewählter oder eingegebener Inhalt der Box.
	DropDownStyle	DropDown erlaubt auch die freie Eingabe eines neuen Wertes, DropDownList erlaubt nur die Auswahl aus der Liste.
	DropDownHeight	Höhe des DropDown in Pixeln. Alternative zu
	MaxDropDownItems	Anzahl der im DropDown darzustellenden Zeilen. Funktioniert nur, wenn IntegralHeight = False ist.
	Items	Gibt den Inhalt des Dropdown zurück. Items (→ ListBox: Methoden).
	Items.Count	Gibt Anzahl der Elemente im Dropdown zurück.
	SelectedIndex	Gibt den (0-basierten) Index des gewählten Elements zurück oder legt ihn fest (z.B. beim Initialisieren der ComboBox). Gibt -1 zurück falls kein Dropdown-Eintrag gewählt ist.
Ereignisse:	TextChanged	Der Inhalt des Eingabefeldes hat sich geändert.
	SelectedIndexChanged	Die Elemente-Auswahl hat sich geändert.
	DoubleClick	Doppelklick auf ein Element (→ lösen Sie die Default-Aktion dazu aus).
Tipps	<ol style="list-style-type: none"> 1. Der Inhalt des DropDowns kann bereits zur Entwurfszeit über den Designer festgelegt werden (Eigenschaft Items). 2. Legen Sie bei einer DropDownList immer einen Start-Selektion fest. Beispiel: <code>cboFood.SelectedIndex = 0;</code> 	

PictureBox

Zweck: Die PictureBox eignet sich hervorragend zur Darstellung von Bildern und Grafiken. Sie eignet sich aber auch als Containerobjekt.

Ein „fixes“ Bild (zur grafischen Illustration in Ihrer Anwendung) fügen Sie dem Programm als PictureBox zu und definieren dabei den Inhalt (Eigenschaft Image) bevorzugt als „Ressource“ (siehe nächstes Kapitel).

Eigenschaften:	(Name)	pic....
	Image	Legt das anzuzeigende Bild fest. Bild zur Design-Zeit festlegen: → Ressource zufügen / wählen. Bild zur Laufzeit laden: picBild.ImageLocation = FileName
	SizeMode	Bestimmt, wie das Bild angezeigt wird:
	Normal	Bild 100%, obere linke Ecke fix.
	StretchImage	Auf die Box skaliert (horizontal und vertikal)
	AutoSize	Passt Box dem Bild an (100%-Darstellung)
	Zoom	Skaliert Bild unter Beibehaltung der Proportionen auf die Grösse der PictureBox.

Tipps

1. Application.StartupPath gibt den Pfad zur .exe-Datei zurück (hilft bei der relativen Angabe der Bilddatei).

Timer

Das Timer-Element finden Sie in der Toolbox in der Gruppe der „Komponenten“. Ein Timer-Element ist im Benutzerinterface nicht sichtbar. Wird ein Timer-Element auf die Form gezogen, so erscheint das Timer-Element in einem speziellen Bereich am unteren Rand des Entwurf-Fensters.

Zweck: Der Timer ist gedacht für Aktionen, die verzögert oder regelmässig in gewissen Zeitabständen ausgeführt werden sollen.

Eigenschaften:	(Name)	tmr....
	Enabled	True Timer läuft und löst nach jedem Ablauf des Intervals das Tick-Ereignis aus. False Timer ist angehalten, löst kein Tick-Ereignis aus.
	Interval	Zeitdauer von einem Tick-Ereignis zum nächsten, in Millisekunden (1000 → 1 Sekunde).
Ereignisse:	Tick	Wird (bei Enabled = True) immer nach Ablauf von Interval ms ausgelöst.

1.9.2 Vorgefertigte Dialogfelder (in der Toolbox ganz unten)

Einige Aufgaben werden in fast allen Windows-Anwendungen gleich gelöst. Dazu gehören etwa die Dialoge zum Öffnen oder Speichern von Dateien, der Druckdialog, sowie die Auswahl von Farben. Für diese Aufgaben stehen vorgefertigte Dialoge zur Verfügung.

Sie finden diese Dialoge in der Toolbox ganz am Ende im Kapitel „Dialogfelder“. Da diese Komponenten normalerweise nicht direkt in der Form angezeigt werden sollen, sondern erst als Reaktion auf eine Benutzeraktion (z.B. auf einen Button-Click) werden sie nicht in der Form dargestellt. Wird ein solcher Standard-Dialog in eine Form gezogen, so erscheint er in einer speziellen Zone am unteren Rand des Entwurf-Fensters.

Als Beispiel: OpenFileDialog

Zweck: Auswahl einer bestehenden Datei.

Eigenschaften:

(Name)	ofd....
FileName	Filename inkl. Pfad und Erweiterung, oder "" (leer) wenn nichts ausgewählt wurde.
Title	Titel des Dialogs
Filter	Filterzeichenfolge für Dateinamen. Bsp: "Text-Files (*.txt) *.txt All files (*.*) *.*"
Initial Directory	Welcher Pfad soll anfänglich eingestellt sein? Bsp: ofdOpenDlg.InitialDirectory = Application.StartupPath
Multiselect	True falls mehrere Dateien ausgewählt werden können.

Methoden:

ShowDialog	Zeigt den Dialog modal und gibt Forms.DialogResult.OK (Datei gewählt) oderCancel (abgebrochen) zurück
------------	--

Verwendung:

Bsp:

```
if (ofdOpenDlg.ShowDialog() == Windows.Forms.DialogResult.OK)
{
    dateiName = ofdOpenDlg.FileName // Dialogdaten auslesen
    ... // Daten verarbeiten
}
```

1.9.3 Zusätzliche Komponenten und Steuerelemente

Weitere Steuerelemente, die nicht standardmässig in der Werkzeugsammlung angezeigt werden, können mit Rechtsklick auf das Toolbox-Fenster → "Choose Items..." hinzugefügt werden.

Achtung: Diese zusätzlichen Steuerelemente sind in Bibliotheks-Dateien abgelegt (.dll/.ocx Dateien). Beim Verteilen der Anwendung müssen diese Dateien unter Umständen mit ausgeliefert werden.

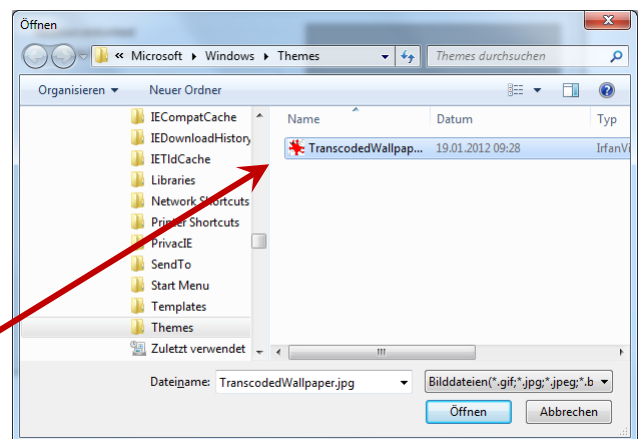
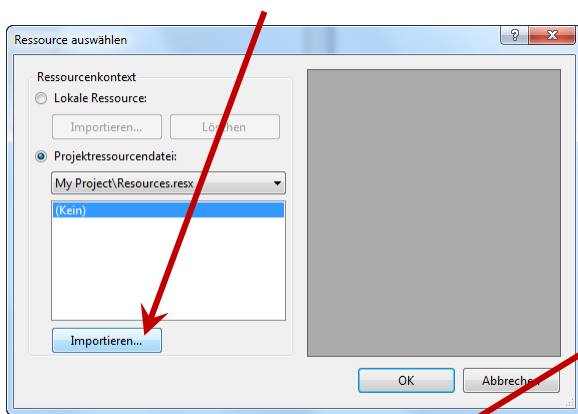
1.10 Ressourcen hinzufügen

Ressourcen sind nicht ausführbare Daten, die logisch mit einer Anwendung bereitgestellt werden. Es sind meist Elemente des Benutzerinterfaces wie Fehlermeldungen, Texte, Bilder, Icons, Cursor und vieles mehr.

Der „binäre“ Inhalt der Ressource wird an den Code des Programms angefügt und sind damit Bestandteil der .exe-Datei. Die Elemente müssen so nicht als externe Dateien mitverteilt werden.

Bsp: Sie wollen Ihrem Dialog ein Hintergrundbild spendieren.

1. Dialog selektieren.
2. Im Eigenschaften-Fenster bei BackGroundImage den ...-Button wählen.
3. Importieren... wählen.



4. Zum gewünschten Bild navigieren und Öffnen wählen.
5. Das als Ressource definierte Bild erscheint nun in der Liste der Projektressourcen.
6. Das Bild ist im Projektmappen-Explorer sichtbar.
7. Bild in der Ressourcenliste selektieren und OK.

