

Wie erstelle ich eine Erweiterung für DevPoint?

Zu aller erst: Was sind Erweiterungen?

- Erweiterungen sind kleine „Programme“, die das Arbeiten mit DevPoint erleichtern sollen

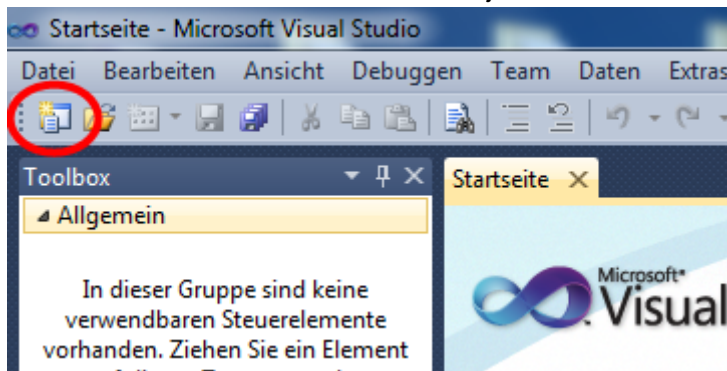
„Ich möchte auch Erweiterungen schreiben, aber wie?“

Man benötigt:

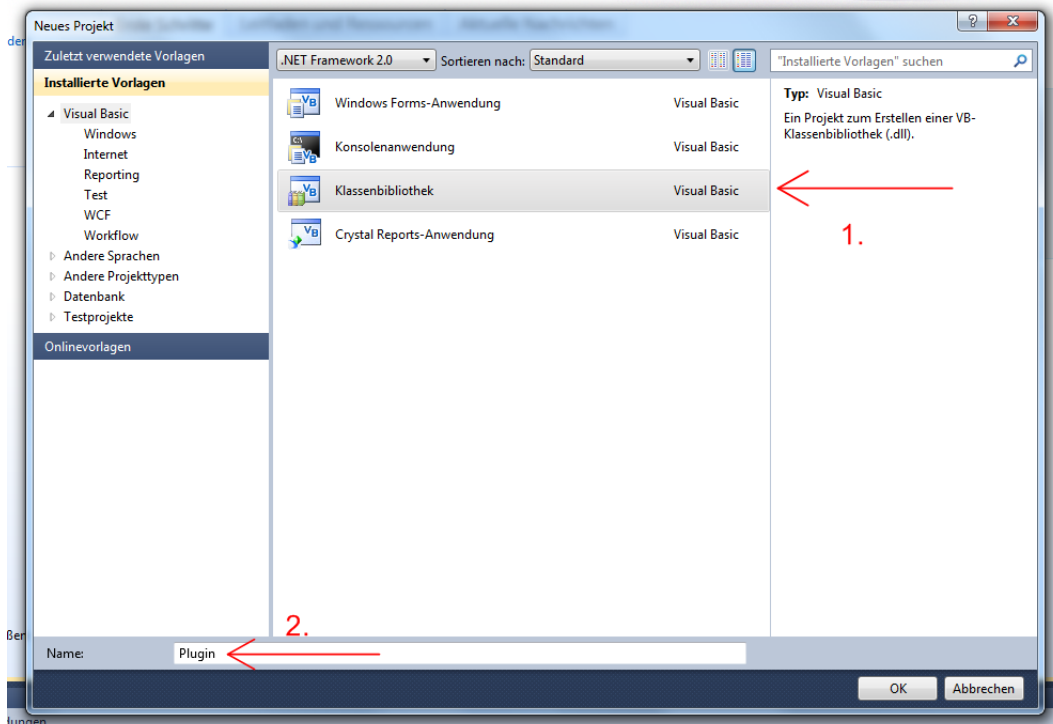
- DevPoint, damit man die Erweiterungen dann auch testen kann
- Eine [Entwicklungsumgebung](#) für eine Sprache des .NET Frameworks. Beispielsweise Visual Basic Express. Auch kann man die Sprachen C# oder Visual C++ benutzen.
- Vorkenntnisse sind sicherlich von Vorteil

Fangen wir nun mal an:

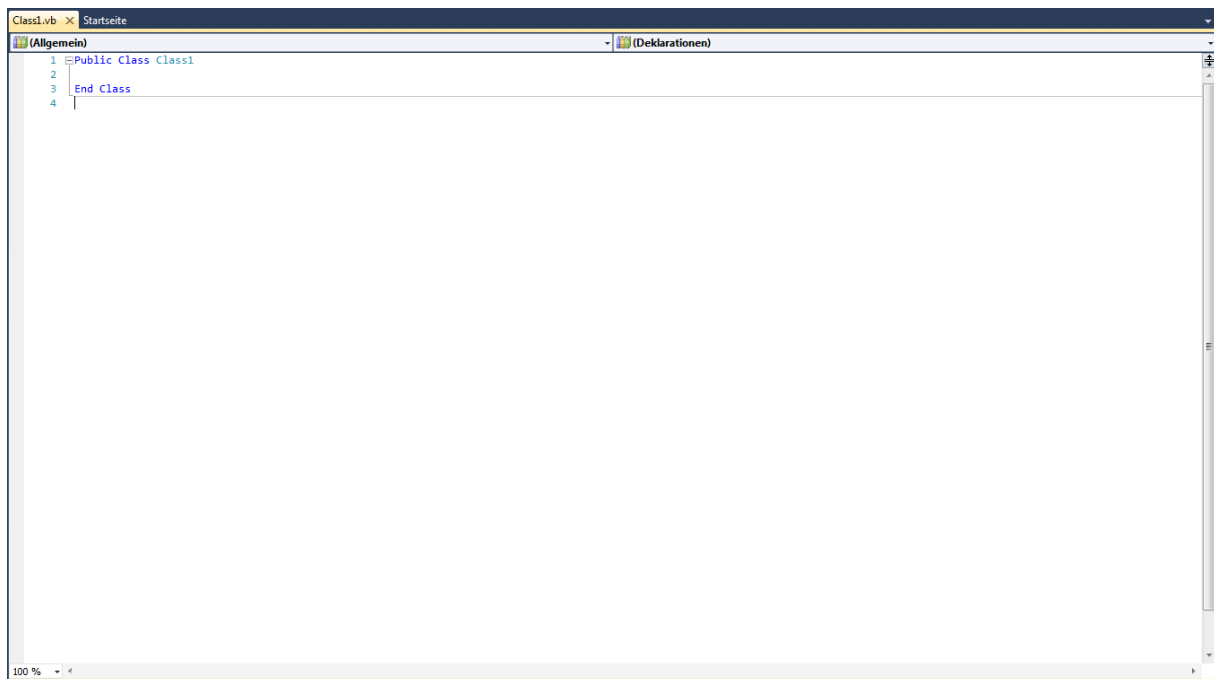
Als erstes erstellen wir ein neues Projekt.



Nun Wählen wir als Projekt „Klassenbibliothek“ aus und nennen das Projekt Plugin.

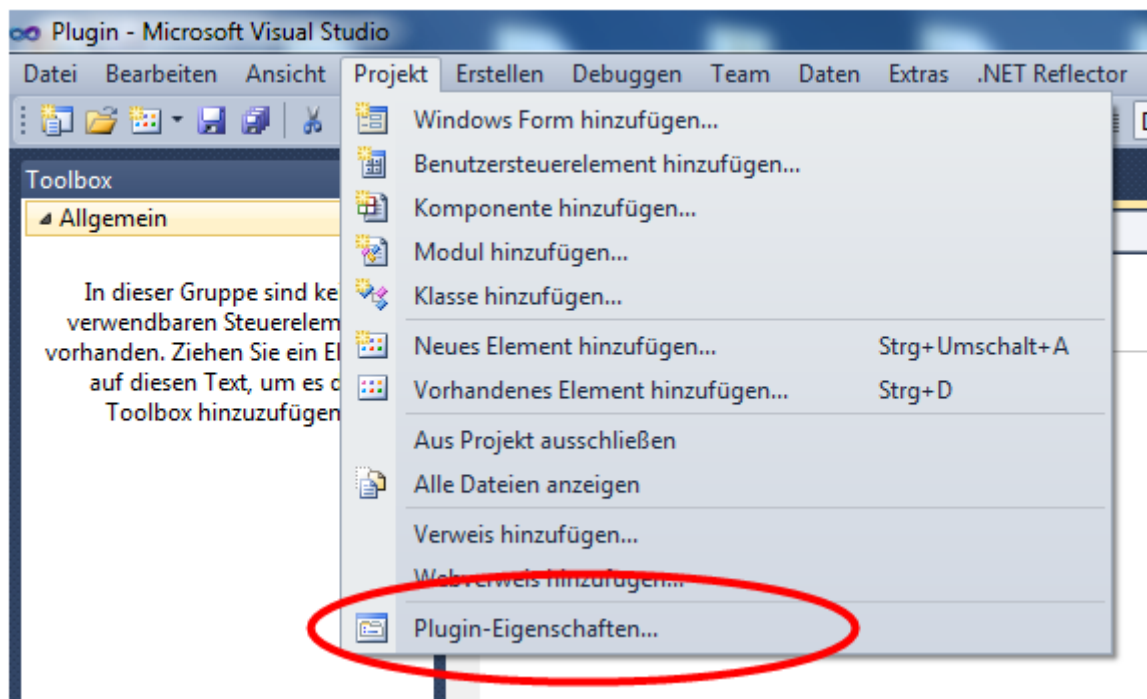


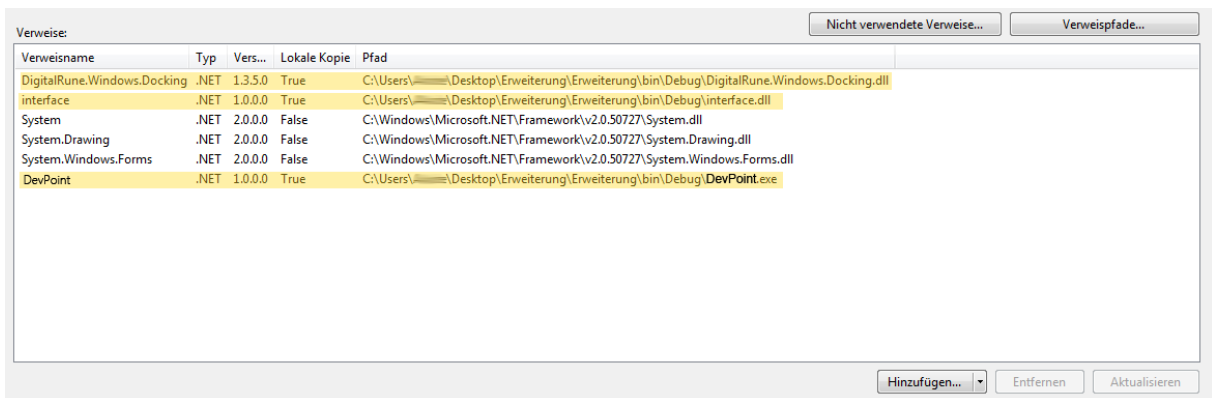
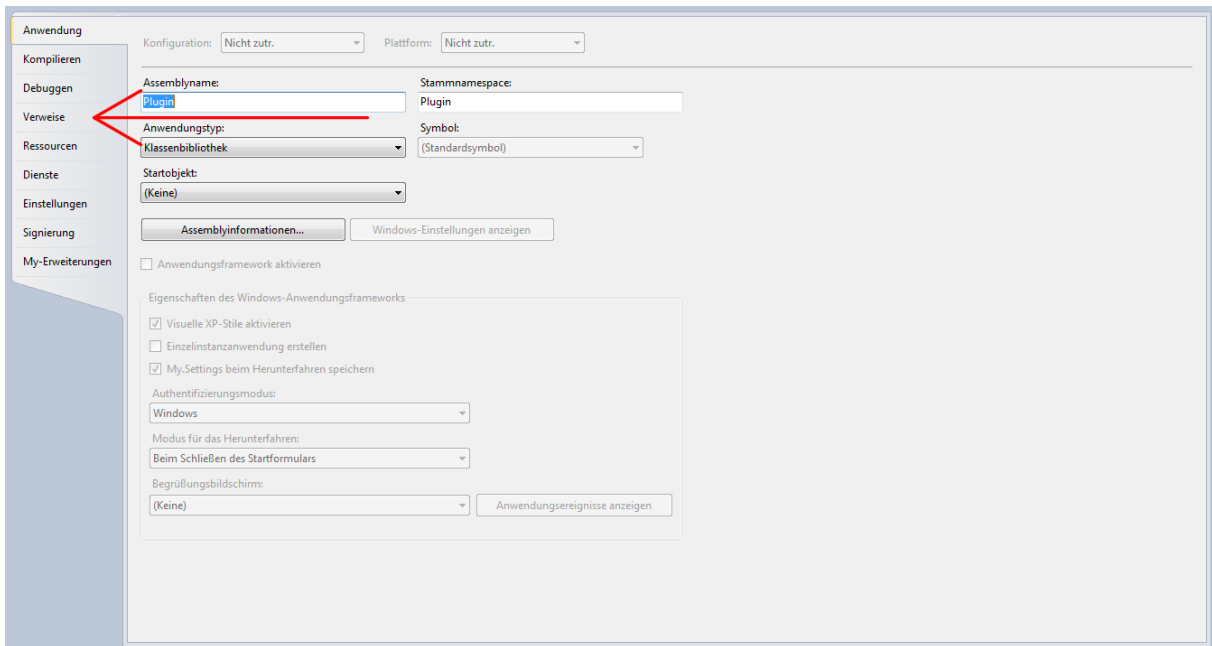
Danach müssten wir folgendes sehen:



```
1 Public Class Class1
2
3 End Class
4
```

Bevor wir nun anfangen zu programmieren, müssen wir noch 3 Verweise hinzufügen, damit die Erweiterung dann später auch funktioniert. Dazu folgen wir folgenden Schritten:





Hier müssen wir Verweise auf folgende Dateien hinzufügen: DevPoint.exe, DigitalRune.Windows.Docking.dll und interface.dll. Dies geschieht über den Button „Hinzufügen...“.

Nachdem wir das getan haben, kommen wir nun zurück zum Quellcode. Wechseln wir also wieder in das Code-Fenster. Dort fügen wir an oberste Stelle vor alles anderem diesen Code ein:

```
Imports System
Imports [interface]
Imports System.Windows.Forms
Imports Devpoint
```

Nun ersetzen wir die Zeile

```
Public Class Class1
```

Durch

```
Friend NotInheritable Class Main
```

Darunter schreiben wir die Implementierung:

```
Implements [interface].pluginsystem
```



Dies gemacht, fügen wir die 4 Grundfunktionen der Erweiterung ein:

```
Public Function getcopyright() As [interface].copyright Implements [interface].pluginsystem.getcopyright
    Dim copy As New [interface].copyright
    copy.name = "Beispiel Plugin"
    copy.version = "1.0"
    copy.Creator = "4TeckWare"
    copy.description = "Dies ist ein einfaches Beispiel"
    Return copy
End Function
```

In dieser Funktion stehen einzelne Informationen zum Copyright und Autor. Diese können Sie bei eigenen Erweiterungen beliebig ändern.

```
Sub closing() Implements [interface].pluginsystem.closing
    MsgBox("Beispiel Plugin beendet!")
End Sub
```

Dies ist das Ereignis, was ausgelöst wird, wenn man DevPoint beendet. Dieses Beispiel lässt eine Meldung mit dem Text „Beispiel Plugin beendet!“ erscheinen.

```
Function getname() As String Implements [interface].pluginsystem.getname
    Return "Pluginname"
End Function
```

Mit dieser Funktion wird der Name der Erweiterung zurückgegeben. Dieser Name steht dann im Erweiterungsmenu von DevPoint.

```
Sub started() Implements [interface].pluginsystem.started
    MsgBox("Beispiel Plugin gestartet!")
End Sub
```

Dies ist das Ereignis, was ausgelöst wird, wenn man DevPoint startet. Dieses Beispiel lässt beim Start von DevPoint eine Meldung mit dem Text „Beispiel Plugin gestartet!“ erscheinen.

Als letzten, grundlegenden Schritt müssen wir den Bereich einfügen, indem der Code steht, der dann beim Klick auf die Erweiterung in DevPoint ausgeführt wird.

```
Sub main() Implements [interface].pluginsystem.main
    Try
        Dim form As New Addieren
        If form Is Nothing Then
            MsgBox("Es trat ein Fehler bei dem Plugin auf. Bitte wenden Sie sich an dessen Entwickler.", MsgBoxStyle.Critical, "Fehler")
        End If
        Devpoint.Form1.AddTab(form, DigitalRune.Windows.Docking.DockState.Document)
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

Erklärung:

Rot: Der rot markierte Bereich fängt Fehler ab (siehe Pink) und gibt sie gegebenenfalls in einer Meldung aus (siehe Blau)

Grün: Der Grün markierte Bereich definiert eine neue Variable namens „form“ als „Addieren“. „Addieren“ ist in diesem Beispiel der Name einer vorher erstellten Form.

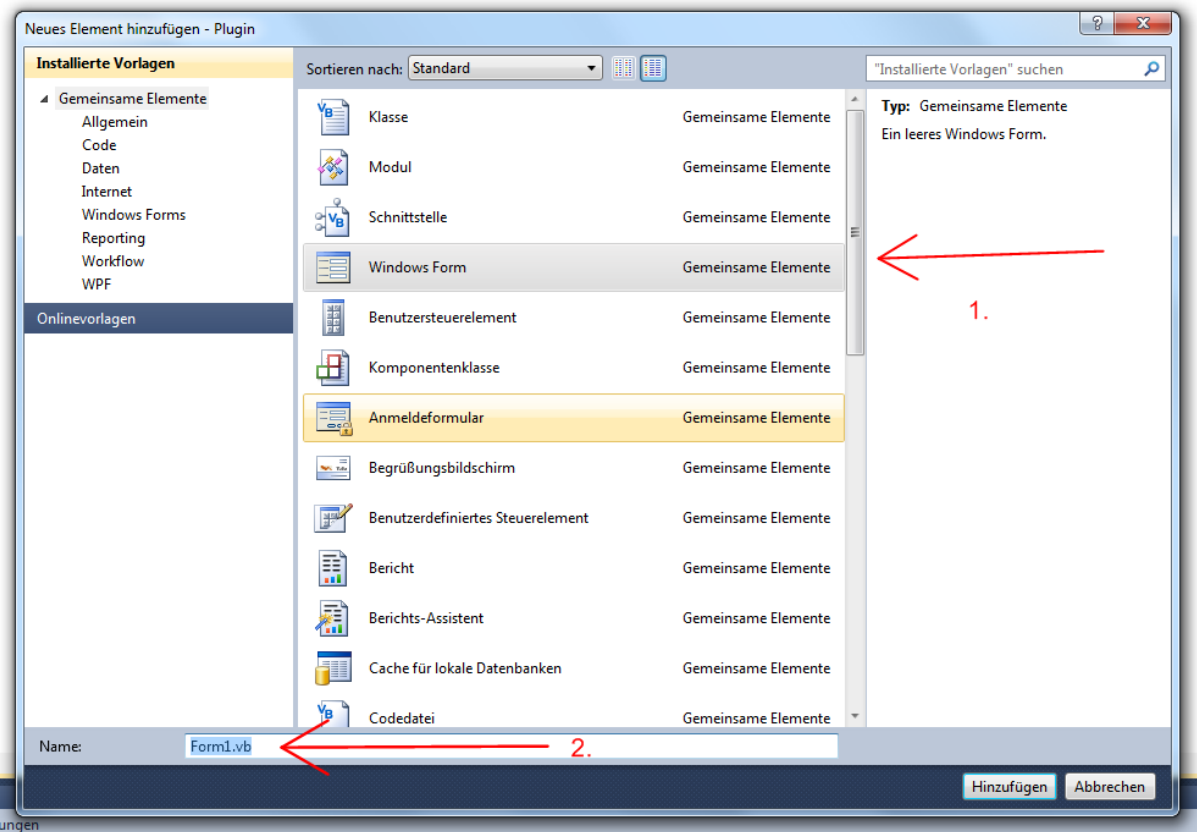
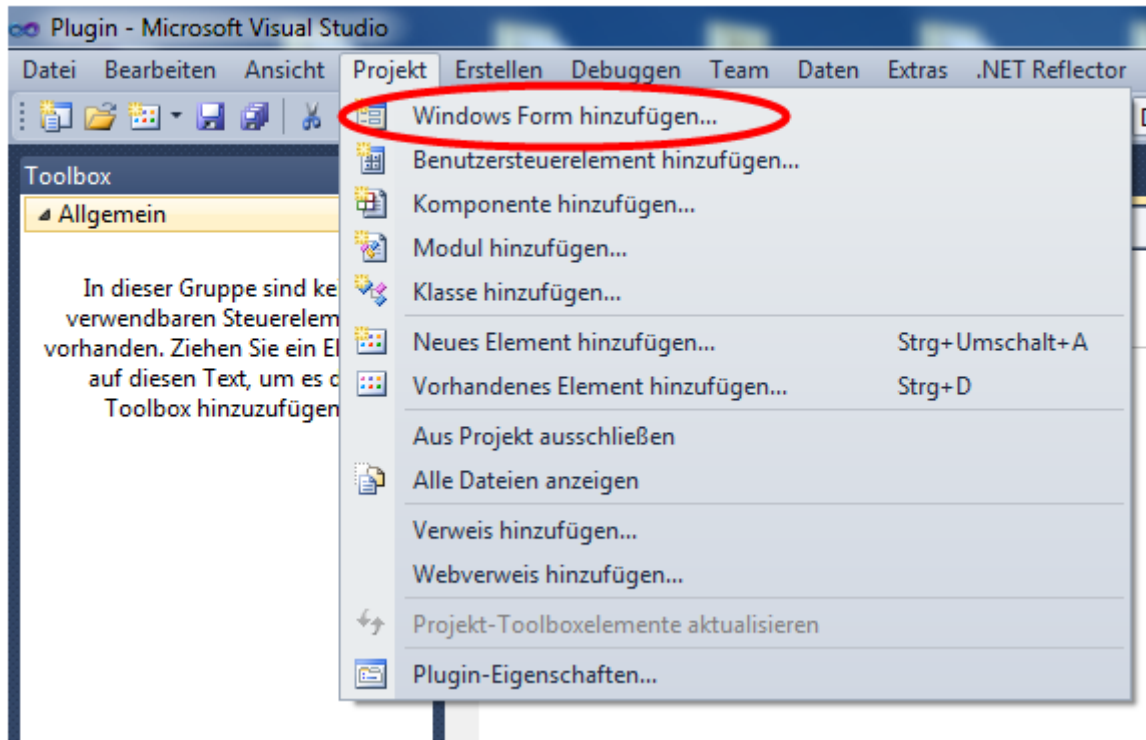
Gelb: Dieser Bereich wird ausgelöst, wenn ein Fehler beim Ausführen aufgetreten ist, weil zum Beispiel eine Datei fehlt oder gar fehlerhaft ist.



Hellblau: Dieser Teil des Codes fügt DevPoint ein neuen Tab hinzu, in dem die Form „Addieren“ dann angezeigt wird.

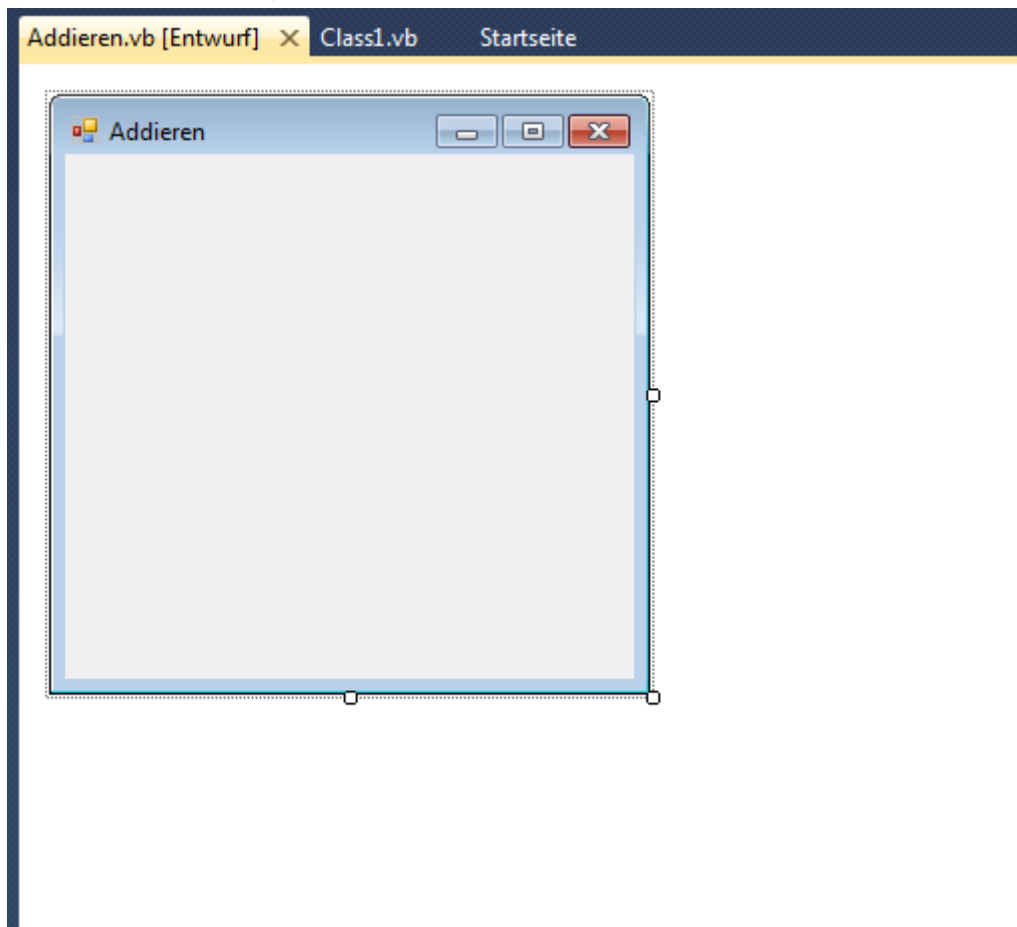
Kommen wir nun zum Schluss zur Erstellung der Form „Addieren“ und deren Programmierung.

Man erstellt eine neue Form wie folgt:



In Schritt 1 wählen wir „Windows Form“ aus. In Schritt 2 tragen wir unten den gewünschten Namen der Form ein. In unserem Beispiel müsste da „Addieren.vb“ stehen statt „Form1.vb“.

Hat alles funktioniert, müsste es so aussehen:



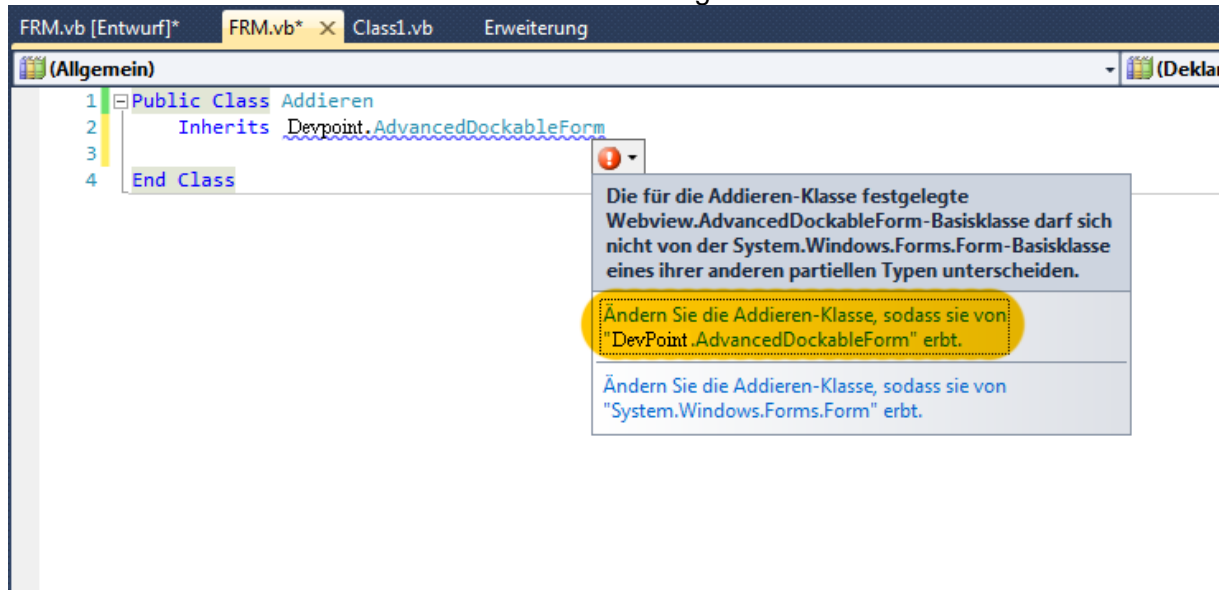
Um zum Codefenster zu gelangen, drücken wir entweder die Taste F7 oder gehen im Menü auf Ansicht -> Code.

Dort schreiben wir in die Klasse „`Inherits DevPoint.AdvancedDockableForm`“, sodass es dann so aussieht:

```
Public Class Addieren
    Inherits DevPoint.AdvancedDockableForm
|
End Class
```

Wie man sehen kann ist hier ein Fehler aufgetreten. Diesen entfernt man, indem man mit dem Mausfeil über das kleine, unter dem „m“, rot erscheinende Rechteck fährt und dann

auf das rote Ausrufezeichen klickt. Nun wählt man folgendes aus:



Der Fehler sollte damit behoben sein. (Der Name „FRM.vb“ oben in der Tableiste auf dem Bild kann ignoriert werden)

Nun kann man in dieser Form ganz normal programmieren, zum Beispiel einen Taschenrechner, was jedoch bei DevPoint eher unpassend wäre.

Auf die Programmierung in der Form wird hier kein Bezug genommen.

