

Visual Basic 2005

Ein Überblick über Sprache und
Entwicklungsumgebung

Übersicht

- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
- Die Entwicklungsumgebung

Eine kurze Einführung in die Welt von VB .NET

- Eine neue, formalisierte Sprache
- Basiert auf „VB Classic“
- Objektorientiert
- Implementiert Reflection
- Baut auf dem umfangreichen .NET-Framework auf
- Behebt viele kleine Probleme von VB 6 ...
- ... bringt aber auch einige mit
- Wird von vielen als „native“ .NET-Sprache gesehen

Übersicht

- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
 - Parametrisierte Typen und Algorithmen („Generik“)
 - Operatorenüberladung
 - Der „My“-Namensbereich
 - Sonstiges
- Die Entwicklungsumgebung

Parametrisierte Typen und Algorithmen

- Was sind parametrisierte Typen?
- Wofür braucht man sie?

Parametrisierte Typen

Bekannt: Typenparameter in C++

```
using std::list;

int main()
{
    list<int> integers;
    integers.push_back(4);
    integers.push_back(2);
    integers.push_back(3);
    integers.push_back("foo"); // Fehler!
}
```

Und jetzt das ganze in VB:

```
Imports System.Collections.Generic

Module Module1
    Sub Main()
        Dim integers As New List(Of Integer)
        integers.Add(4)
        integers.Add(2)
        integers.Add(3)
        integers.Add("foo") ' Fehler!
    End Sub
End Module
```

Parametrisierte Typen

Eigene parametrisierte Klassen schreiben:

```
Class FixedStack(Of T)
    Private data As T()
    Private offset As Integer = 0

    Public Sub New(ByVal size As Integer)
        ReDim Me.data(size - 1)
    End Sub

    Public Sub Push(ByVal value As T)
        Me.data(Me.offset) = value
        Me.offset += 1
    End Sub

    Public Function Pop() As T
        Me.offset -= 1
        Return Me.data(Me.offset)
    End Function
End Class

Module Module1
    Sub Main()
        Dim test As New FixedStack(Of Char)(5)
        test.Push("a"c)
        test.Push("b"c)
        Console.WriteLine(test.Pop())
        Console.WriteLine(test.Pop())
    End Sub
End Module
```

Parametrisierte Algorithmen

Auch Algorithmen lassen sich parametrisieren:

```
Module Module1
    Private Sub CreateAndDispose(Of T As {IDisposable, New})()
        Using x As New T()
            End Using
        End Sub

    Private Class Debug
        Implements IDisposable

        Public Overloads Sub Dispose() Implements IDisposable.Dispose
            Console.WriteLine("Bye, bye, babe ...")
        End Sub
    End Class

    Sub Main()
        CreateAndDispose(Of Debug)()
    End Sub
End Module
```

Parametrisierte Algorithmen

Auch Algorithmen lassen sich parametrisieren:

```
Module Module1
    Private Sub CreateAndDispose(Of T As {IDisposable, New})()
        Using x As New T()
        End Using
    End Sub

    Private Class Debug
        Implements IDisposable

        Public Overloads Sub Dispose() Implements IDisposable.Dispose
            Console.WriteLine("Bye, bye, babe ...")
        End Sub
    End Class

    Sub Main()
        CreateAndDispose(Of Debug)()
    End Sub
End Module
```

Typenparameter besitzen Kontext („type constraints“)

Parametrisierte Algorithmen

Auch Algorithmen lassen sich parametrisieren:

```
Module Module1
    Private Sub CreateAndDispose(Of T As {IDisposable, New})()
        Using x As New T()
        End Using
    End Sub

    Private Class Debug
        Implements IDisposable

        Public Overloads Sub Dispose() Implements IDisposable.Dispose
            Console.WriteLine("Bye, bye, babe ...")
        End Sub
    End Class

    Sub Main()
        CreateAndDispose(Of Debug)()
    End Sub
End Module
```

Typenparameter besitzen Kontext („type constraints“)

Typen implementieren diesen Kontext

Parametrisierte Algorithmen

Auch Algorithmen lassen sich parametrisieren:

```
Module Module1
  Private Sub CreateAndDispose(Of T As {IDisposable, New})()
    Using x As New T()
    End Using
  End Sub

  Private Class Debug
    Implements IDisposable

    Public Overloads Sub Dispose() Implements IDisposable.Dispose
      Console.WriteLine("Bye, bye, babe ...")
    End Sub
  End Class

  Sub Main()
    CreateAndDispose(Of Debug)()
  End Sub
End Module
```

Typenparameter besitzen Kontext („type constraints“)

Typen implementieren diesen Kontext

Neues Sprachmerkmal: der „Using“-Block

Parametrisierte Typen und Algorithmen

- Was sind parametrisierte Typen?
- Wofür braucht man sie?
 - Mehr Informationen zur Compilezeit
 - Typensicherheit
 - Effizienz

Übersicht

- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
 - Parametrisierte Typen und Algorithmen („Generik“)
 - Operatorenüberladung
 - Der „My“-Namensbereich
 - Sonstiges
- Die Entwicklungsumgebung

Operatorenüberladung

Ist eigentlich nichts neues:

```
Dim a As Double = 1.0 + 2.4  
Dim b As String = "a" + "b"
```

Der „+“-Operator führt hier, je nach Parametertypen, unterschiedliche Funktionen aus (Addition von Gleitkommazahlen, Stringkonkatenation).

Operatorenüberladung

Ist eigentlich nichts neues:

```
Dim a As Double = 1.0 + 2.4  
Dim b As String = "a" + "b"
```

Der „+“-Operator führt hier, je nach Parametertypen, unterschiedliche Funktionen aus (Addition von Gleitkommazahlen, Stringkonkatenation).

Aber das hier ging bisher nicht:

```
Class Complex  
End Class  
  
Sub Main()  
    Dim c As New Complex(1.0, 2.0)  
    Dim i As New Complex(0.0, 1.0)  
    Console.WriteLine("i + c = {0}", i + c) ' Addition „Complex“ mit „Complex“  
    Console.WriteLine("1 + i = {0}", 1 + i) ' Addition „Integer“ mit „Complex“  
End Sub
```

Operatorenüberladung

Unter VB 2005 kann man diesen Code mit sehr geringem Aufwand zum Laufen bringen:

```
Partial Class Complex
    Public Shared Operator +(ByVal lhs As Complex, ByVal rhs As Complex) As Complex
        Return New Complex(lhs.Re + rhs.Re, lhs.Im + rhs.Im)
    End Operator

    Public Shared Operator +(ByVal lhs As Double, ByVal rhs As Complex) As Complex
        Return New Complex(lhs + rhs.Re, rhs.Im)
    End Operator
End Class
```

Operatorenüberladung

Unter VB 2005 kann man diesen Code mit sehr geringem Aufwand zum Laufen bringen:

```
Partial Class Complex
    Public Shared Operator +(ByVal lhs As Complex, ByVal rhs As Complex) As Complex
        Return New Complex(lhs.Re + rhs.Re, lhs.Im + rhs.Im)
    End Operator

    Public Shared Operator +(ByVal lhs As Double, ByVal rhs As Complex) As Complex
        Return New Complex(lhs + rhs.Re, rhs.Im)
    End Operator
End Class
```

Definition eines „+“-Operators über zwei „Complex“-Parameter

Operatorenüberladung

Unter VB 2005 kann man diesen Code mit sehr geringem Aufwand zum Laufen bringen:

```
Partial Class Complex
    Public Shared Operator +(ByVal lhs As Complex, ByVal rhs As Complex) As Complex
        Return New Complex(lhs.Re + rhs.Re, lhs.Im + rhs.Im)
    End Operator

    Public Shared Operator +(ByVal lhs As Double, ByVal rhs As Complex) As Complex
        Return New Complex(lhs + rhs.Re, rhs.Im)
    End Operator
End Class
```

Definition eines „+“-Operators über zwei „Complex“-Parameter

Operatoren sind „Shared“

Operatorenüberladung

Unter VB 2005 kann man diesen Code mit sehr geringem Aufwand zum Laufen bringen:

```
Partial Class Complex
    Public Shared Operator +(ByVal lhs As Complex, ByVal rhs As Complex) As Complex
        Return New Complex(lhs.Re + rhs.Re, lhs.Im + rhs.Im)
    End Operator

    Public Shared Operator +(ByVal lhs As Double, ByVal rhs As Complex) As Complex
        Return New Complex(lhs + rhs.Re, rhs.Im)
    End Operator
End Class
```

Definition eines „+“-Operators über zwei „Complex“-Parameter

Operatoren sind „Shared“

Operatoren müssen in Klassen definiert sein

Operatorenüberladung

Unter VB 2005 kann man diesen Code mit sehr geringem Aufwand zum Laufen bringen:

```
Partial Class Complex
    Public Shared Operator +(ByVal lhs As Complex, ByVal rhs As Complex) As Complex
        Return New Complex(lhs.Re + rhs.Re, lhs.Im + rhs.Im)
    End Operator

    Public Shared Operator +(ByVal lhs As Double, ByVal rhs As Complex) As Complex
        Return New Complex(lhs + rhs.Re, rhs.Im)
    End Operator
End Class
```

Definition eines „+“-Operators über zwei „Complex“-Parameter

Operatoren sind „Shared“

Operatoren müssen in Klassen definiert sein

Mindestens einer der Parameter muss vom Typ der Klasse sein

Operatorenüberladung

Unter VB 2005 kann man diesen Code mit sehr geringem Aufwand zum Laufen bringen:

```
Partial Class Complex
    Public Shared Operator +(ByVal lhs As Complex, ByVal rhs As Complex) As Complex
        Return New Complex(lhs.Re + rhs.Re, lhs.Im + rhs.Im)
    End Operator

    Public Shared Operator +(ByVal lhs As Double, ByVal rhs As Complex) As Complex
        Return New Complex(lhs + rhs.Re, rhs.Im)
    End Operator
End Class
```

Definition eines „+“-Operators über zwei „Complex“-Parameter

Operatoren sind „Shared“

Operatoren müssen in Klassen definiert sein

Mindestens einer der Parameter muss vom Typ der Klasse sein

Neues Sprachmerkmal: Partielle Klassen („Partial“-Schlüsselwort)

Konversionsoperatoren

Konversionen sind implizite oder explizite Operatorenaufrufe.

```
Partial Class Complex
    Public Shared Narrowing Operator CType(ByVal value As Complex) As Double
        Return Math.Sqrt(value.Re ^ 2 + value.Im ^ 2) ' Vektorbetrag
    End Operator

    Public Shared Widening Operator CType(ByVal value As Double) As Complex
        Return New Complex(value, 0.0)
    End Operator
End Class

Module Module1
    Sub Main()
        Dim c As New Complex(1.0, 2.0)
        Dim n As Complex = 1 ' Implizite Konversion
        Dim v As Double = CDb1(c) ' Explizite Konversion
    End Sub
End Module
```

Wahrheitswert-Operatoren

Spezialfall der Konversion: Umwandlung in Wahrheitswert

```
Partial Class Complex
    Public Shared Operator IsTrue(ByVal value As Complex) As Boolean
        Return value IsNot Nothing AndAlso (value.Re <> 0 OrElse value.Im <> 0)
    End Operator

    Public Shared Operator Not(ByVal value As Complex) As Boolean
        If value Then Return False Else Return True
    End Operator

    Public Shared Operator IsFalse(ByVal value As Complex) As Boolean
        Return Not value
    End Operator
End Class

Module Module1
    Sub Main()
        Dim i As New Complex(0.0, 1.0)
        If i Then MsgBox("i hat einen wert ungleich null.")
    End Sub
End Module
```

Übersicht

- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
 - Parametrisierte Typen und Algorithmen („Generik“)
 - Operatorenüberladung
 - Der „My“-Namensbereich
 - Sonstiges
- Die Entwicklungsumgebung

Der „My“-Namensbereich

- Bietet schnellen Zugriff auf wichtige Funktionen
- Ist klar gruppiert und übersichtlich
- Enthält Zugriff auf anwendungsspezifische Werte
- Arbeitet im Kontext des aktiven Benutzers
- Schafft laut lesbaren, intuitiv verständlichen Code, zum Beispiel:

```
' „show a message box with my user name“  
MsgBox(My.User.Name)  
  
' „play an explosion from my resources on on my computer/audio device  
' and wait for it to complete“  
My.Computer.Audio.Play(My.Resources.explosion, AudioPlayMode.waitToComplete)
```

„My.Application“

- Objekt existiert im Namensbereich „My“
- Anders als „System.Windows.Forms.Application“
- Gruppiert Informationen zur eigenen Anwendung
- Innerhalb der Klasse Reaktion auf App-Events möglich
 - „NetworkAvailabilityChanged“
 - „Shutdown“
 - „Startup“
 - „StartupNextInstance“
 - „UnhandledException“

„My.Computer“

- Zugriff auf häufig benötigte Funktionen des Systems
- Informationen über den PC und das System
- Zugriff auf Hardware-Schnittstellen

„My.Forms“

- Liste aller im Projekt enthaltenen Form-Klassen
- *Nicht nur* bereits geöffnete Forms (wie in VBC)
- Ähnlich der impliziten Form-Erstellung in VBC
- Erstellt bei Zugriff streng typisierte Instanz
- „My.Forms.“ kann weggelassen werden

„My.Resources“

- Kein Objekt sondern Namensbereich in „My“
- Streng typisierte Properties für alle Ressourcen
- Wrapper für internen Ressourcenmanager
- Arbeitet im aktuellen länderspezifischen Kontext

„My.Settings“

- Objekt existiert im Namensbereich „My“
- Lädt (neu), speichert, aktualisiert Einstellungen
- Einstellungen sind streng typisierte Properties
- Einstellungen sind lokal (Benutzer) oder global (alle)
- Innerhalb der Klasse Reaktion auf Events möglich
 - „PropertyChanged“
 - „SettingChanging“
 - „SettingsLoaded“
 - „SettingsSaving“

„My.User“

- Informationen über den aktuellen Benutzer
 - Name
 - Rolle (Admin, Gast, ...)
- = „Microsoft.VisualBasic.ApplicationServices.User“

Übersicht

- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
 - Parametrisierte Typen und Algorithmen („Generik“)
 - Operatorenüberladung
 - Der „My“-Namensbereich
 - Funktionsobjekte
 - Sonstiges
- Die Entwicklungsumgebung

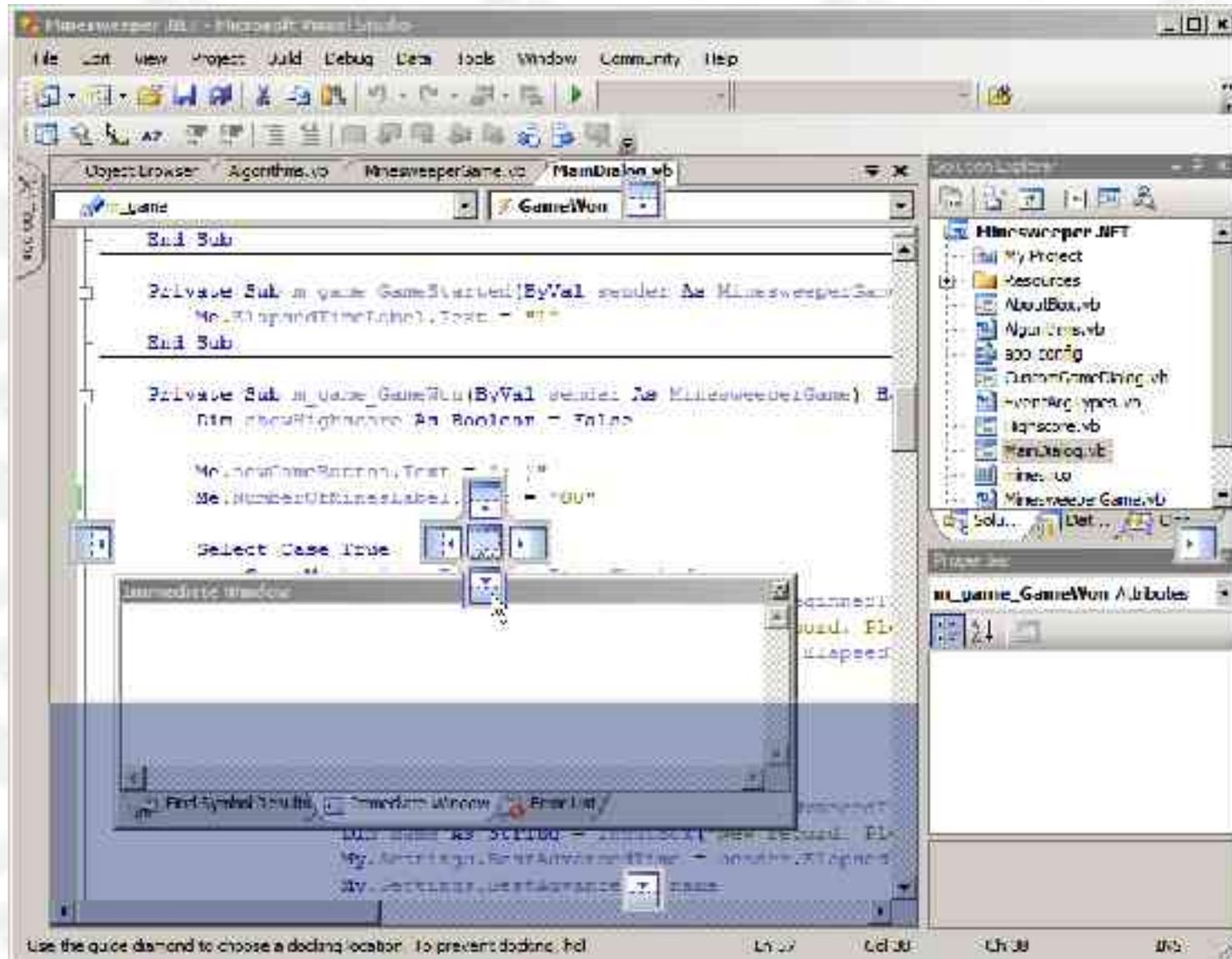
Sonstiges

- VB kann mit vorzeichenlosen Ganzzahltypen arbeiten
- Neuer Typ „System.Nullable(Of T As Structure)“
- Neues „Continue“-Schlüsselwort in Schleifen
- Neues „Global“-Schlüsselwort als Root-Namensbereich
- Property Get/Set kann unterschiedliche Sichtbarkeitsbereiche haben
- „IDisposable“-Schnittstelle verbessert
- Enumeratoren-Schnittstellen verbessert
- XML-Kommentare unterstützt

Übersicht

- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
- **Die Entwicklungsumgebung**
 - „IDE Docking Control“
 - IntelliSense, reloaded
 - Form Designer
 - Debug-Werkzeuge
 - Codeschnipsel

„IDE Docking Control“



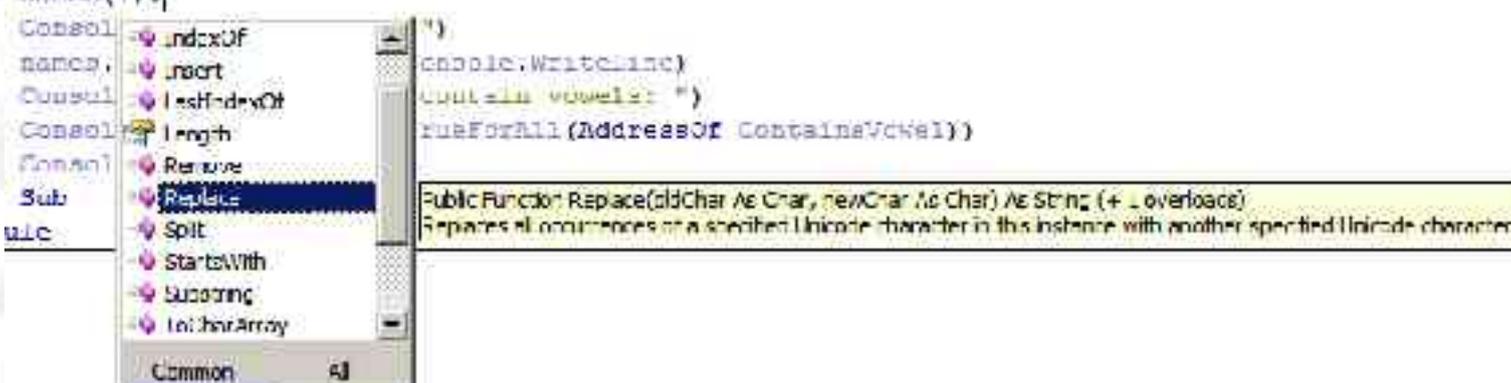
Übersicht

- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
- **Die Entwicklungsumgebung**
 - „IDE Docking Control“
 - IntelliSense, reloaded
 - Form Designer
 - Debug-Werkzeuge
 - Codeschnipsel

IntelliSense, reloaded

Anzeigen von Datentypen und Objektmitgliedern mittels IntelliSense

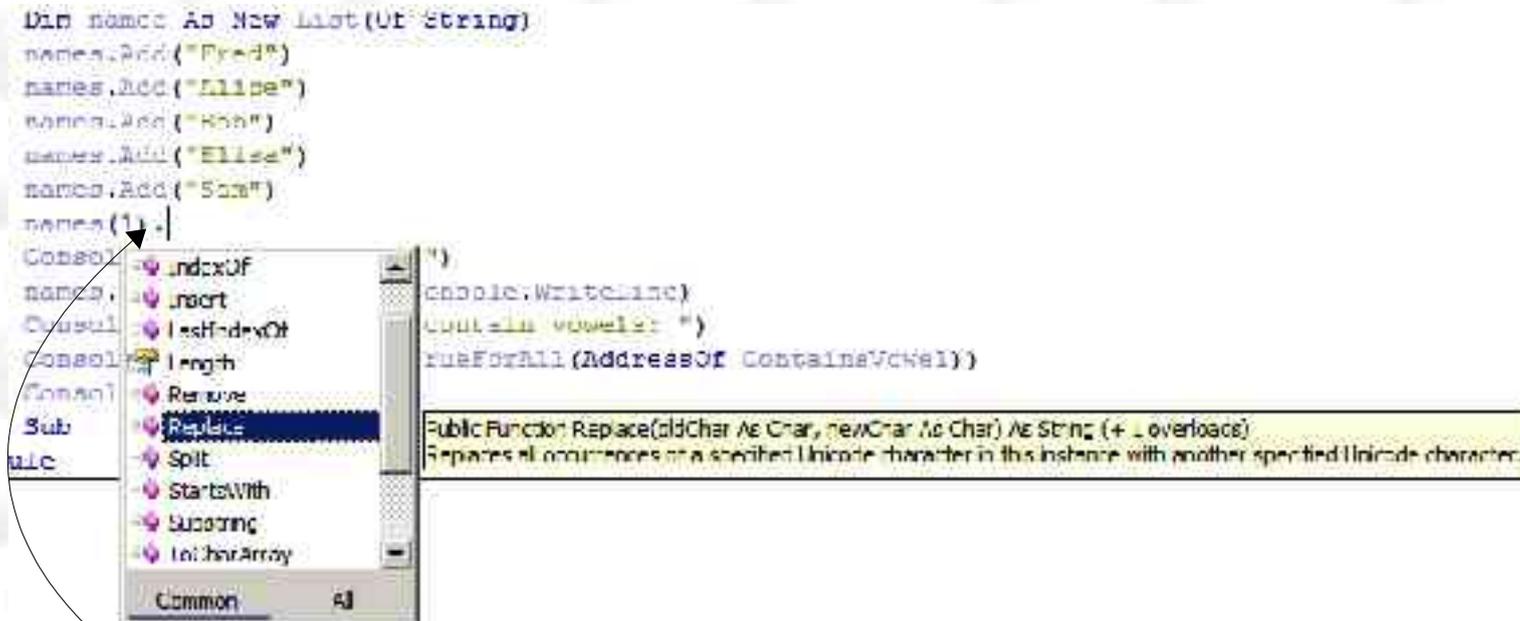
```
Dim names As New List(Of String)
names.Add("Fred")
names.Add("Elise")
names.Add("Ron")
names.Add("Elise")
names.Add("Sam")
names(1).|
```



indexOf	Console.WriteLine()
insert	Console.WriteLine()
indexOf	Console.WriteLine()
indexOf	Console.WriteLine()
length	ForEachAll(AddressOf ContainsVowel))
remove	
replace	Public Function Replace(oldChar As Char, newChar As Char) As String (+ 1 overloads) Replaces all occurrences of a specified Unicode character in this instance with another specified Unicode character.
split	
startsWith	
substring	
toCharArray	

IntelliSense, reloaded

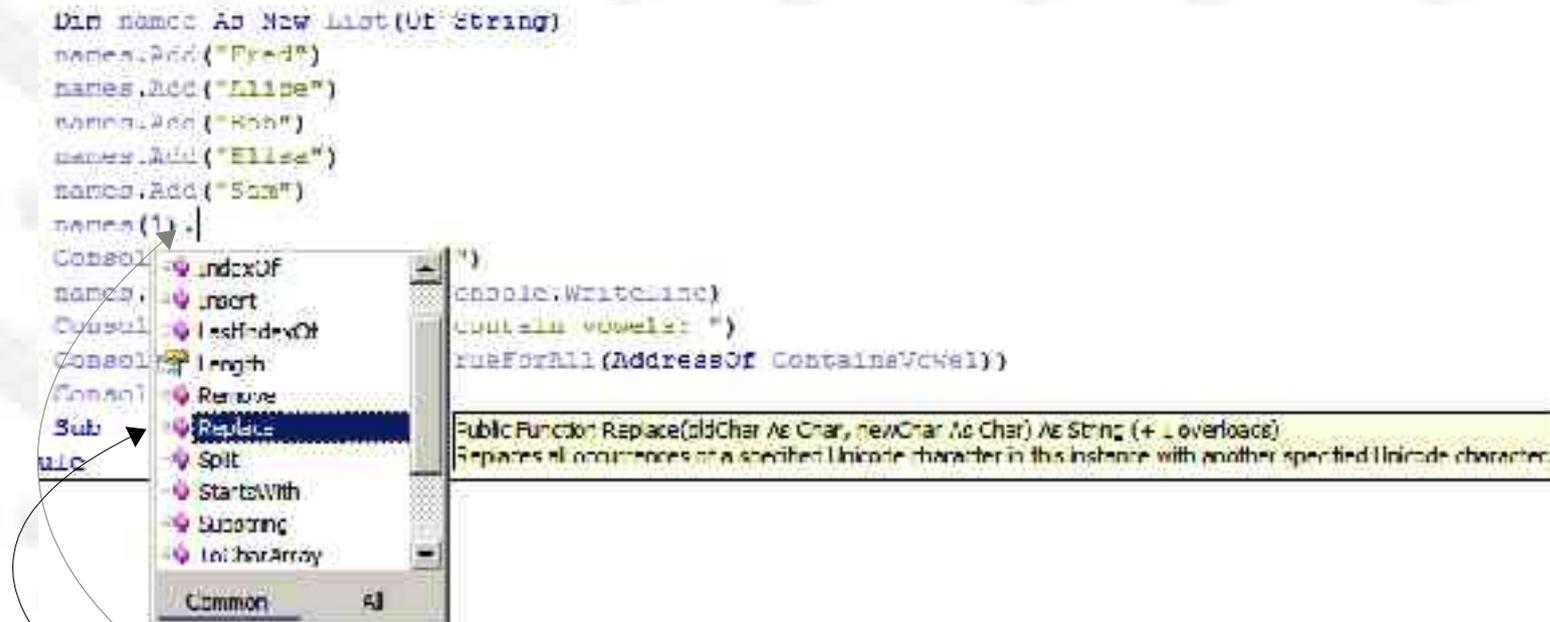
Anzeigen von Datentypen und Objektmitgliedern mittels IntelliSense



Generischer Typ der Elemente wird erkannt

IntelliSense, reloaded

Anzeigen von Datentypen und Objektmitgliedern mittels IntelliSense



Generischer Typ der Elemente wird erkannt

Häufig benötigtes Element wird automatisch fokussiert

IntelliSense, reloaded

Anzeigen von Datentypen und Objektmitgliedern mittels IntelliSense

```
Dim names As New List(Of String)
names.Add("Fred")
names.Add("Elise")
names.Add("Ron")
names.Add("Elise")
names.Add("Sam")
names(1) = |
```

Informationen über Parameter und Verwendung

Public Function Replace(oldChar As Char, newChar As Char) As String (+ 1 overloads)
Replaces all occurrences of a specified Unicode character in this instance with another specified Unicode character.

Generischer Typ der Elemente wird erkannt

Häufig benötigtes Element wird automatisch fokussiert

Selten verwendete Elemente können ausgeblendet werden

IntelliSense, reloaded

Automatische Fehlerkorrekturvorschläge

```
Dim names As New List(Of String)
Dim ages As New List(Of Integer)
names.Add("Eve")
names.Add("Alice")
names.Add("Bob")
names.Add("Ella")
names.Add("Sam")
Console.WriteLine("Names: ")
names.ForEach(AddressOf Console.WriteLine)
```



The screenshot shows a code editor window with the following code:

```
Dim names As New List(Of String)
Dim ages As New List(Of Integer)
names.Add("Eve")
names.Add("Alice")
names.Add("Bob")
names.Add("Ella")
names.Add("Sam")
Console.WriteLine("Names: ")
names.ForEach(AddressOf Console.WriteLine)
```

An IntelliSense error correction menu is displayed over the code, showing the error: "Type 'Integer' is not defined." The menu offers four options:

- Change Integer to 'Integer'
- Change Integer to 'Int32'
- Change Integer to 'Int16'
- Change Integer to 'Int64'

IntelliSense, reloaded

Refactor! In Aktion

```
''' <summary>  
''' Shuffles a list of values.  
''' </summary>  
''' <typeparam name="T">Type of the objects in list.</typeparam>  
''' <param name="list">Input list.</param>  
''' <returns>Returns a LIFO stack which contains the shuffled values.</returns>  
Public Function Shuffle(Of T) (ByVal list As List(Of T)) As Stack(Of T)  
    Dim ret As New Stack(Of T)  
    Dim end As Integer = list.Count - 1  
    While list.Count > 0  
        Dim index As Integer = end  
        ret.Push(list.ElementAt(index))  
        list.RemoveAt(index)  
    End While  
  
    Return ret  
End Function
```



The screenshot shows the Visual Studio IDE with the 'Shuffle' function signature selected. A context menu is open over the function signature, with 'Reorder Parameters' highlighted. A tooltip for 'Reorder Parameters' is displayed, explaining that it reorders parameters in the current method and updates calling code to reflect the new order.

Übersicht

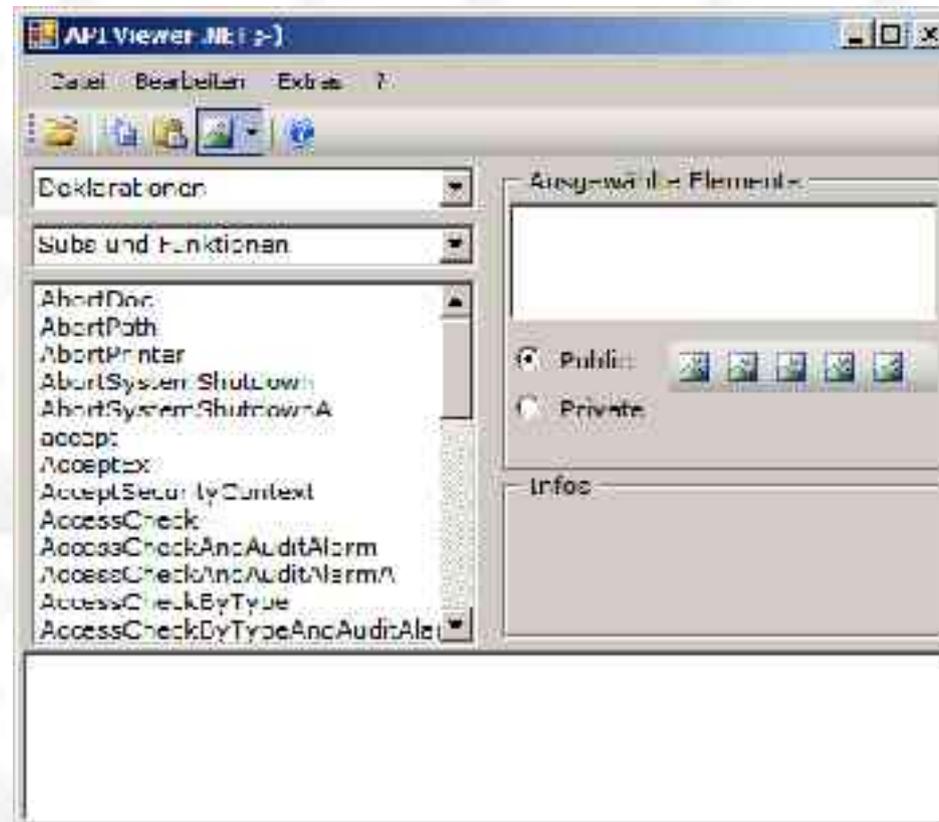
- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
- **Die Entwicklungsumgebung**
 - „IDE Docking Control“
 - IntelliSense, reloaded
 - **Form Designer**
 - Debug-Werkzeuge
 - Codeschnipsel

Form Designer

- Blitzschnelles Erstellen professioneller Dialoge
- Moderne Windows Forms Controls
- Integrierter Editor für Menüs und Toolstrips
- Häufige Aufgaben in kleinen Popups zusammengefasst
- Erleichtertes Gruppieren durch Containercontrols
- Erleichtertes Ausrichten durch intelligente Hilfslinien
- Viele fertige Dialogvorlagen benutzbar

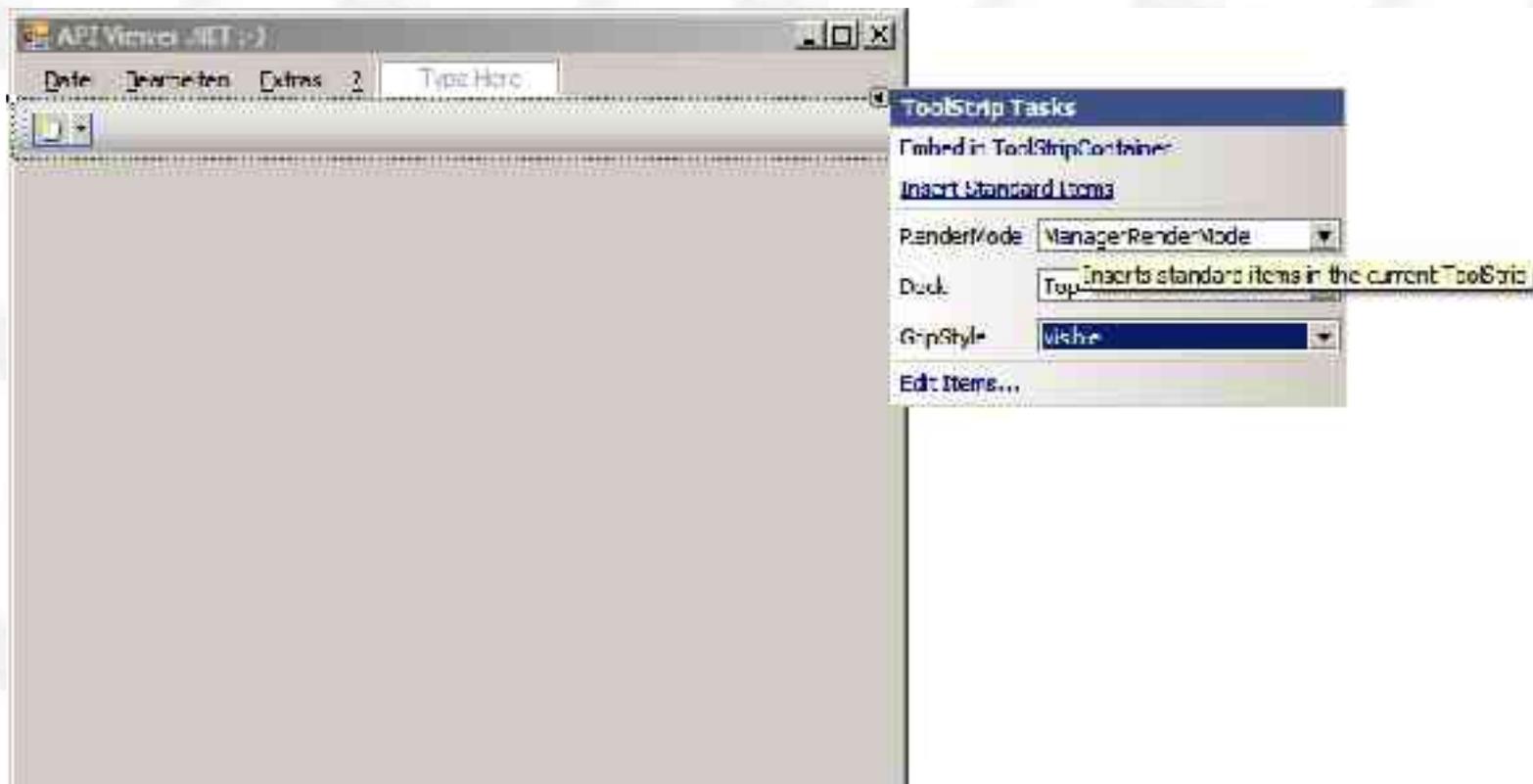
Form Designer

Ein „fertiger“ Dialog



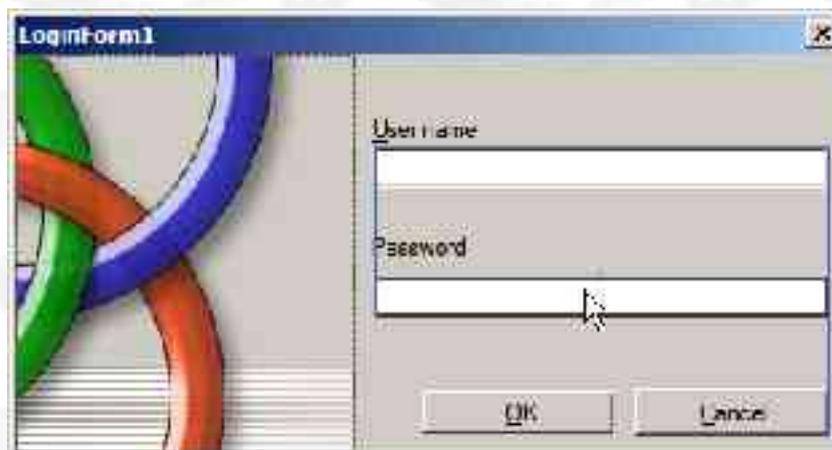
Form Designer

ToolTips für häufige Aufgaben



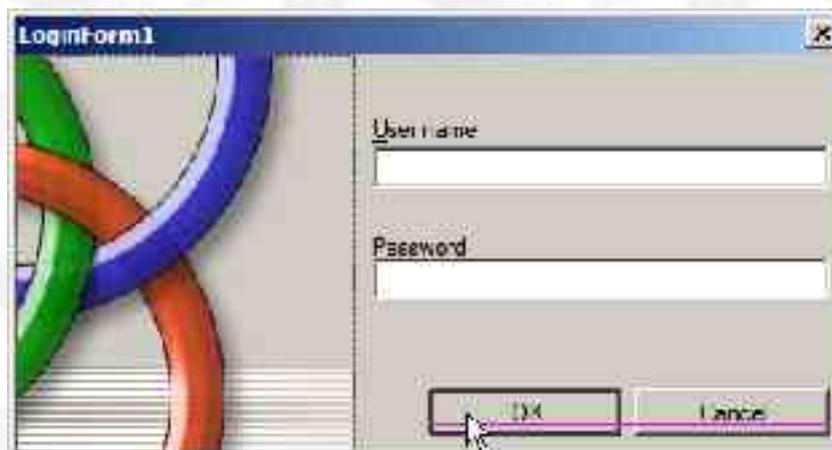
Form Designer

Ausrichten von Controls an SnapLines



Form Designer

Ausrichten von Controls an Textbasislinien



Übersicht

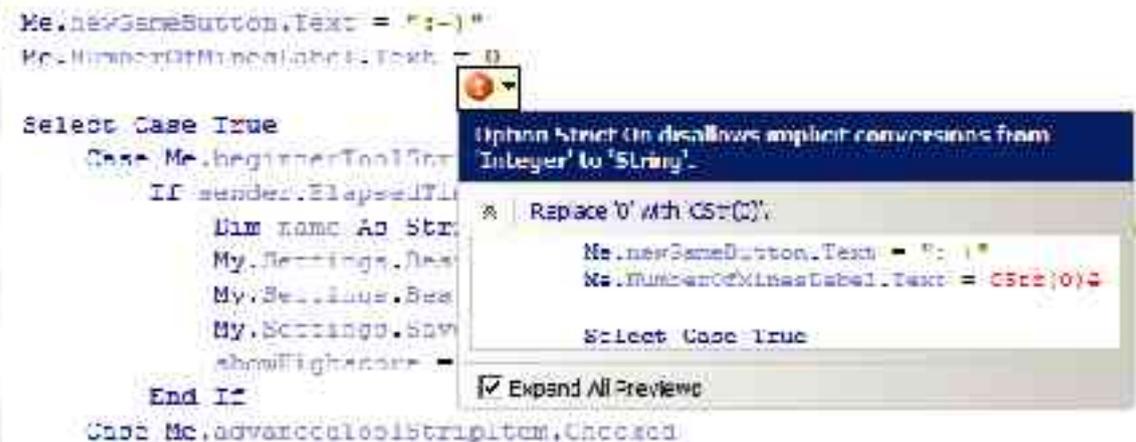
- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
- **Die Entwicklungsumgebung**
 - „IDE Docking Control“
 - IntelliSense, reloaded
 - Form Designer
 - **Debug-Werkzeuge**
 - Codeschnipsel

Debug-Werkzeuge

Automatische Fehlerkorrekturhilfen

```
Me.newGameButton.Text = "(-)"
Me.NumberOfMinasLabel.Text = 0

Select Case True
  Case Me.beginnerToolTip
    If sender.ElapsedTime < 10000 Then
      Dim name As String
      My.Settings.DefaultName = name
      My.Settings.DefaultLevel = 1
      My.Settings.DefaultNumberOfMinas = 1
      My.Settings.DefaultNumberOfWalls = 1
      My.Settings.DefaultNumberOfWalls = 1
      showHighscore = True
    End If
  Case Me.advancedToolTip
    Checked = True
```



Upon Strict (It disallows implicit conversions from 'Integer' to 'String').

Replace '0' with 'CS(C)'.
Me.newGameButton.Text = "(-)"
Me.NumberOfMinasLabel.Text = CS(C)0

Select Case True

Expand All Previews

Debug-Werkzeuge

Verbesserte Überwachung von Werten

```
ElseIf action = MouseActionType.MouseUp Then
    ' right button up
    ChangeFieldState (Me.m_field(FieldID:V1, FieldID:V2))
    DoGraphicRefresh = True Me.m_field (length=2)
End If
End If
Gamestate.Paused
GameState.Lost
GameState.Won
ect

event event post-processing

ction = MouseButton.Left Then
action = MouseActionType.MouseUp Then
Me.m_lastButton = Me.m_lastButton
if
-Button = MouseButton.Right Then
action = MouseActionType.MouseUp Then
Me.m_lastButton = Me.m_lastButton
If
```



(0,0)	{macrat.Minesweeper.MinesweeperGame.Field}	
DoGraphicUpdate	False	MinesweeperGame.Field
Number	0	MinesweeperGame.Field
State	Hidden {0}	MinesweeperGame.Field
Value	Value {1}	MinesweeperGame.Field
(0,2)	{macrat.Minesweeper.MinesweeperGame.Field}	
(0,0)	{macrat.Minesweeper.MinesweeperGame.Field}	
(0,1)	{macrat.Minesweeper.MinesweeperGame.Field}	
(0,8)	{macrat.Minesweeper.MinesweeperGame.Field}	
(1,0)	{macrat.Minesweeper.MinesweeperGame.Field}	
(1,1)	{macrat.Minesweeper.MinesweeperGame.Field}	
(1,2)	{macrat.Minesweeper.MinesweeperGame.Field}	
(1,3)	{macrat.Minesweeper.MinesweeperGame.Field}	
(1,4)	{macrat.Minesweeper.MinesweeperGame.Field}	
(1,5)	{macrat.Minesweeper.MinesweeperGame.Field}	

Debug-Werkzeuge

Verbesserte Überwachung von Werten

```
Select Case True
  Case Me.beginnerToolStripItem.Checked
    My.Settings.LastGameType = "Beginner"
  Case Me.advancedToolStripItem.Checked
    My.Settings = {macrat.Minesweeper.Settings}
  My.Settings
  Case Me.expertToolStripItem.Checked
    My.Settings
  Case Me.customToolStripItem.Checked
    My.Settings
  My.Settings.Key
  My.Settings.AddedHandler
  My.Settings.BestAdvanced
  My.Settings.BestAdvancedTime
  My.Settings.BestBeginner
  My.Settings.BestBeginTime
  My.Settings.BestExpert
  My.Settings.BestExpertTime
  My.Settings.Context
  My.Settings.Default
  My.Settings.DefaultInstance
  My.Settings.HighScoreToolStripItem.Checked
    My.Settings.HighScore = New HighScoreToolStripItem

End Select

My.Settings.Save()
End Sub

Private Sub MainDialog_Load(object sender, EventArgs e)
  Me.Show()
End Sub
```

Property	Value
_Properties	Nothing
_PropertyValues	{System.Configuration.SettingsPropertyValueCollection}
_providers	{System.Configuration.SettingsProviderCollection}
_settings	{System.Configuration.SettingsPropertyCollection}
settingsKey	""
addedHandler	True
BestAdvanced	"(unknown)"
BestAdvancedTime	99
BestBeginner	"korrad"
BestBeginTime	12
BestExpert	"(unknown)"
BestExpertTime	999
Context	Count = 3
Default	{macrat.Minesweeper.Settings}
defaultInstance	{macrat.Minesweeper.Settings}

Debug-Werkzeuge

Umfangreiche Informationen bei Laufzeitfehlern

```
Public Sub New(ByVal graphicsDevice As PictureBox, ByVal timer As Windows.Forms.Timer)
    Throw New InvalidOperationException(
        "The operation performed on " & graphicsDevice.Name & " is invalid."
    )
    Me.m_graphicsDevice = graphicsDevice
    Me.m_timer = timer
End Sub
```

InvalidOperationException was unhandled

The operation performed on plowhid is invalid.

Troubleshooting tips:

Get general help for this exception.

Search for more Help Online...

Actions:

- View Detail...
- Copy exception detail to the clipboard

```
Public Sub New
    ByVal m_timer As Integer
End Sub
```

Übersicht

- Eine kurze Einführung in die Welt von VB .NET
- Die Sprache VB 2005
- **Die Entwicklungsumgebung**
 - „IDE Docking Control“
 - IntelliSense, reloaded
 - Form Designer
 - Debug-Werkzeuge
 - Codeschnipsel

Codeschnipsel

- Direkt in die IDE integrierte Codeschnipsel
- Einfügen über Kontextmenü oder Tastenkombination
- Simples Navigieren durch Verwaltungshierarchie
- Erweiterbar (Auch aus dem Internet)
- Schnipsel sind in XML-Format abgelegt
- Ersetzbare „Literale“ im Schnipselcode
- Für verschiedene Sprachen verfügbar (VB, C#, XML ...)

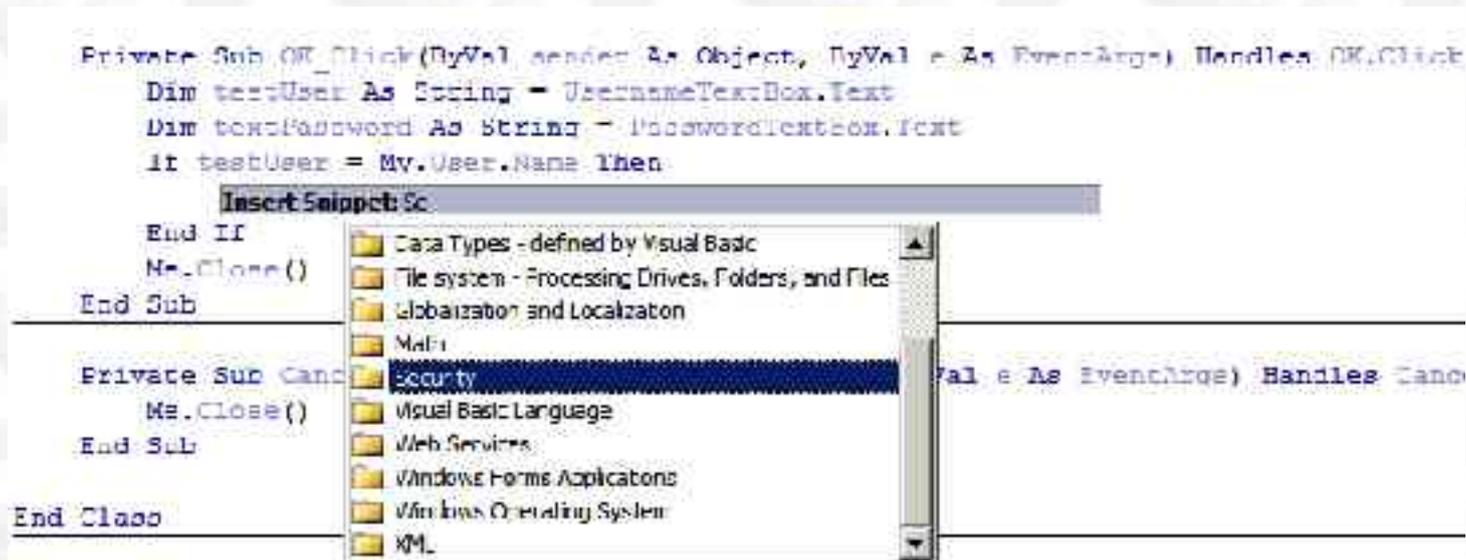
Codeschnipsel

Eine minimale Schnipseldatei

```
<?xml version="1.0" encoding="UTF-8"?>
<CodeSnippets xmlns="http://schemas.microsoft.com/visualstudio/2005/CodeSnippet">
  <CodeSnippet Format="1.0.0">
    <Header>
      <Title>Calculate a square root</Title>
      <Author>Konrad Ludwig Moritz Rudolph</Author>
    </Header>
    <Snippet>
      <Imports>
        <Import>
          <Namespace>System</Namespace>
        </Import>
      </Imports>
      <Declarations>
        <Literal Editable="true">
          <ID>Number</ID>
          <Type>Double</Type>
          <Default>2.0</Default>
        </Literal>
      </Declarations>
      <Code Language="VB" Kind="method body">
        Dim result As Double = Math.Sqrt($Number$)
      </Code>
    </Snippet>
  </CodeSnippet>
</CodeSnippets>
```

Codeschnipsel

Einfügen eines Schnipsels in den Code



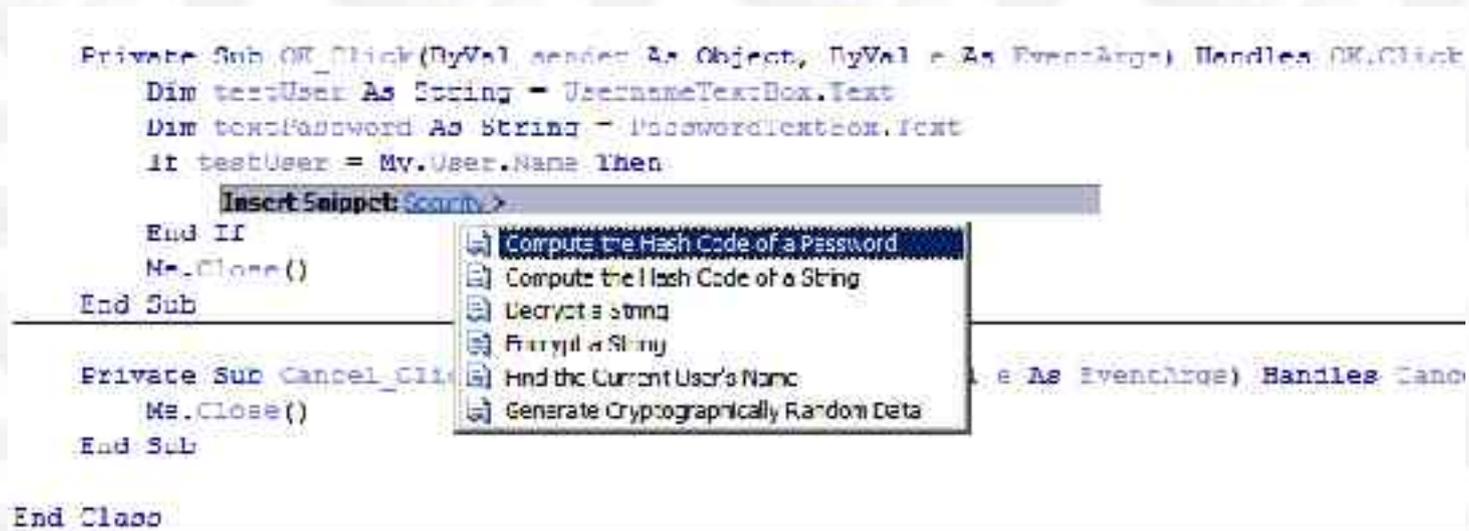
Codeschnipsel

Einfügen eines Schnipsels in den Code

```
Private Sub OK_Click(ByVal sender As Object, ByVal e As EventArgs) Handles OK.Click
    Dim testUser As String = UsernameTextBox.Text
    Dim testPassword As String = PasswordTextbox.Text
    If testUser = My.User.Name Then
        Insert Snippet: Security >
        End If
        Me.Close()
    End Sub

Private Sub Cancel_Click(ByVal sender As EventArgs) Handles Cancel.Click
    Me.Close()
End Sub

End Class
```



Codeschnipsel

Einfügen eines Schnipsels in den Code

```
Private Sub OK_Click(ByVal sender As Object, ByVal e As EventArgs) Handles OK.Click
    Dim testUser As String = UsernameTextBox.Text
    Dim testPassword As String = PasswordTextbox.Text
    If testUser = My.User.Name Then
        Dim sampleEncoder As Encoder = Unicode.GetEncoder()
        Dim md5 As MD5CryptoServiceProvider = New MD5CryptoServiceProvider()
        Dim password As String = testPassword
        Dim bytes(password.Length * 2) As Byte

        sampleEncoder.GetBytes(password.ToCharArray, 0, password.Length, bytes, 0, True)

        Dim hash() As Byte = md5.ComputeHash(bytes)
    End If
    MD.Close()
End Sub
```

Verweise und Bibliographie

- PDF-Version zum Herunterladen
 - <http://activevb.de/members/konradr/workshop05/vortrag.pdf>
- Bibliographie
 - *Introducing Visual Basic 2005 for Developers* (2005) - Microsoft Press
 - *VB .NET Language in a Nutshell* (2001) - O'Reilly
 - WeFly247, *C# 2.0 Language Enhancements* (2005) - Tony Whitter
 - <http://msdn.microsoft.com/vbasic/default.aspx?pull=/library/en-us/dnvs05/html/vbmy.asp>