Continue

Home and Learn: VB Net Programming Course Contents This Home and Learn computer course is an introduction to Visual Basic .NET programming for beginners. This course assumes that you have no programming experience whatsoever. It's a lot easier than you think, and can be a very rewarding hobby! (Once you have completed the beginner's course, we have lots of Intermediate Programming projects and walkthroughs for you. Click the menu on the left for more details.) You don't need to buy any software for this course! You can use the free Visual Basic Community Edition from Microsoft. To see which version you need, click below: Getting the Visual Studio - Which version do I need? Give programming a go - it's easier than you think! VB .NET - The Next Step We have lots of Intermediate Programming Projects for you, now that you have completed the beginners course. You can find them on their own page by clicking the link to the right, or the menu on the left. VB Intermediate Programming (9 Projects) The substring function is used to obtain a part of a specified string. This method is defined in the String class of Microsoft VB.NET. You have to specify the start index from which the String will be extracted. The String will be extracted from that index up to the length that you specify. In this tutorial, you will learn Syntax of Substring The function accepts two arguments as shown in the following syntax: Public Function Substring(ByVal start_Index As Integer, ByVal sub_length As Integer) As String Here, The ByVal keyword denotes pass-by-value, which is a mechanism of passing arguments to functions. The start_Index is the index from which the substring will be obtained sub_length denotes the length up to which the String will be copied from the start_Index. This length is measured in terms of the number of characters. The function will return the extracted substring. Examples Step 1) Create a new console application. Step 2) Add the following code to it: Module Module1 Sub Main() Dim st As String = "Guru99" Dim subst As String = st.Substring(0, 4) Console.WriteLine("The substring is: {0}", subst) Console.ReadKey() End Sub End Module Step 3) Click the Start button from the toolbar to execute the code. You should get the following result: We have used the following code: Explanation of Code: Creating a module named Module1. Starting the main sub-procedure. Defining a string variable named st and assigning the value Guru99 to it. Defining a string variable named ‘subst’ as a substring of the String ‘st’ from index 0f and a length of 4 characters. Printing some text and the above substring on the console. Pausing the console window for a while waiting for the user to take action to close it. End of the main sub-procedure. End of the module. One Argument What if we pass only one argument to the function? The function will copy all the data in the String that begins from that index. What happens is that the Substring function internally copies all the string data at that index as well as that which follows that index. For example: Module Module1 Sub Main() Dim st As String = "Guru99" Dim subst As String = st.Substring(4) Console.WriteLine("The substring is: {0}", subst) Console.ReadKey() End Sub End Module Click the Start button to run the code. It should return the following: The substring function returned 99. We passed the parameter 4 to the function, meaning that it will begin to extract the substring from the character at index 4 to the end of the String. 9 is the character at index 4 of the string Guru99, hence the extraction started there. Middle Characters It is also possible for us to get the middle characters of the String in question. In this case, we only have to provide the starting index and the length of the String that we need. In the following example, we are getting a substring of the specified String from index 2 and the String will have a length of 2 characters: Module Module1 Sub Main() Dim st As String = "Guru99" Dim subst As String = st.Substring(2, 2) Console.WriteLine("The substring is: {0}", subst) Console.ReadKey() End Sub End Module Click the Start button from the toolbar to run the code. You will get the following result: In the above example, the substring function returned ru. We passed the parameters (2, 2) to the function. The first 2 instructs the function to begin the extraction of the substring from index 2 while the second 2 instructs the function to return a substring with a length of 2 characters only. This means that the extraction of the substring should begin from the element located at index 2 of the string Guru99, which is r. Since the returned substring should only have a length of 2 characters, the extraction will not go past the ‘u’, hence it returned ‘ru.’ One Char We can use the Substring function to get a single character from a string. In such a case, it is a necessity for you to make an allocation but the character can be accessed directly. This is a bit faster. The following example demonstrates two ways through which we can achieve this: Module Module1 Sub Main() Dim st As String = "Guru99" Dim mid1 As Char = st(1) Console.WriteLine(mid1) Dim mid2 As String = st.Substring(1, 1) Console.WriteLine(mid2) Console.ReadKey() End Sub End Module Click the Start button to run the code. You will get the following result: We have used the following code: Explanation of Code: Creating a module named Module1. Starting the main sub-procedure. Defining a string variable named st and assigning the value Guru99 to it. Defining a string variable named mid1 and getting the character at index 1 of String st. This character will be assigned to the variable mid1. Printing the above character on the console. Defining a string variable named mid2 and getting the character at index 1 with a length of 1 from String st. The length of 1 means that it will return the same character at the starting index. The counting begins from the starting index that you specify. This character will be assigned to the variable mid2. Printing the above character on the console. Pausing the console window for a while waiting for the user to take action to close it. End of the main sub-procedure. End of the module. Summary The Substring function is defined in the String class of Visual Basic.NET. It accepts two arguments, which is the starting point of the substring and the length of the substring. We can play around with these arguments to get various sets of substrings from the main String. This VB.NET tutorial is a step-by-step guide to learn Visual Basic programming. This free Visual Basic tutorial covers topics like Arrays, Strings, Operators, Switch, Loops, etc. This VB .NET tutorial will help you learn VB.NET basics and advanced concepts. What should I know? This online VB tutorial guide is designed for beginners to learn VB.NET concepts. Refer this link to install Visual Studio. VB.NET Training Syllabus Introduction Advanced Stuff Must Know! The combobox control helps you to display a drop-down list with many items. See it as a combination of a textbox in which a user enters text and a dropdown list from which a user selects an item. Note that the combobox shows one item at a time. Creating a Combobox A ComboBox can be created as follows: Step 1) Create a new Application. Step 2) Drag a combobox control from the toolbox to the form. You will have created a combobox control. Adding Items to Combobox Now that we have created a combobox, let us demonstrate how to add items to it. Double click the combobox control that you have added. You will be moved from the design tab to the tab with code. To add an item to a combobox control, we use the Items property. Let us demonstrate this by adding two items to the combobox, Male and Female: ComboBox1.Items.Add("Male") ComboBox1.Items.Add("Female") We can also choose to add items to the combobox at design time from the Properties window. Here are the steps: Step 1) Open the design tab and click the combobox control. Step 2) Move to the Properties window and view the Items option. Step 3) Click the … located to the right of (Collection). Step 4)You will see a new window. This is where you should add items to the combobox, as shown below: Step 5) Once done with typing the items, click the OK button. Step 6) Click the Start button from the top toolbar and click the dropdown icon on the combobox. The items were successfully added to the combobox control. Selecting Combobox Items You may need to set the default item that will be selected when the form is loaded. You can achieve this via the SelectedItem() method. For example, to set the default selected gender to Male, you can use the following statement: ComboBox1.SelectedItem = "Male" When you run the code, the combobox control should be as shown below: Retrieving Combobox Values You can get the selected item from your combobox. This can be done using the text property. Let us demonstrate this using our above combobox with two items that is, Male and Female. Follow the steps given below: Step 1) Double click the combobox to open the tab with VB.NET code. Step 2) Add the following code: Public Class Form1 Private Sub ComboBox1_SelectedIndexChanged(sd As Object, evnt As EventArgs) Handles ComboBox1.SelectedIndexChanged Dim var_gender As String var_gender = ComboBox1.Text MessageBox.Show(var_gender) End Sub End Class Step 3) Click the Start button from the toolbar to execute the code. You should get the following form: Step 4) Click the dropdown button and choose your gender. In my case. I choose Male, and I get the following: Here is a screenshot of the code: Explanation of Code: Creating a class named Form1. The class will be publicly accessible since its access modifier has been set to Public. Starting of a sub-procedure named ComboBox1_SelectedIndexChanged. This is generated automatically when you double click the combobox control from the design tab. This sub-procedure will be invoked when you select an item from the combobox. The sd As Object references the object that raised the event while the event As EventArgs has the event data s. Creating a string integer named var_gender. Setting the value of variable var_gender to the item that is selected on the combobox. Printing the value of the variable var_gender on a MesageBox. End of the ComboBox1_SelectedIndexChanged sub-procedure. End of the Form1 class. Removing Combobox Items It is possible for you to remove an item from your combobox. There are two ways through which you can accomplish this. You can use either the item index or the name of the item. When using the item index, you should use the Items.RemoveAt() property as shown below: ComboBox1.Items.RemoveAt(1) In the above example, we are removing the item located at index 1 of the combobox. Note that combobox indexes begin at index 0, meaning that the above command will remove the second item of the combobox. To remove the item using its name, you should use the Items.Remove() property as shown below: ComboBox1.Items.Remove("Female") The above code should remove the item named Female from the ComboBox1. Binding DataSource A ComboBox can be populated from a Dataset. Consider the SQL Query given below: select emp_id, emp_name from employees; You can create a datasource in a program then use the following code to bind it: comboBox1.DataSource = ds.Tables(0) comboBox1.ValueMember = "emp_id" comboBox1.DisplayMember = "emp_name" This will provide you with an easy way of populating your combobox control with data without having to type each individual item. SelectedIndexChanged event This type of event is invoked when you change the selected item on your combobox. It is the event you should use when you need to implement an action upon a change on the selected item of a combobox. Let us demonstrate this by use of an example: Step 1) Create a new Window Forms Application. Step 2) After that you need to Drag and drop two combobox controls into the form. Step 3) Double click inside the form to open the tab for code. Enter the following code: Public Class Form1 Private Sub Form1_Load(sd As Object, evnt As EventArgs) Handles MyBase.Load ComboBox1.Items.Add("Males") ComboBox1.Items.Add("Females") End Sub Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ComboBox1.SelectedIndexChanged ComboBox2.Items.Clear() If ComboBox1.SelectedItem = "Males" Then ComboBox2.Items.Add("Nicholas") ComboBox2.Items.Add("John") ElseIf ComboBox1.SelectedItem = "Females" Then ComboBox2.Items.Add("Alice") ComboBox2.Items.Add("Grace") End If End Sub End Class Step 4) Click the Start button from the top bar to run the code. You should get the following output: Step 5) Click the dropdown button on the first combobox and choose Male. Move the mouse cursor to the second combobox and click its dropdown button. See the available items: Step 6) Move to the first combobox and choose Female. Move to the second combobox and see the available items: Here is a screenshot of the code: Explanation of Code: Creating a class named Form1. Start of a sub-procedure named Form1_Load(). This will be triggered once the form is loaded. The sd As Object references the object that raised the event while the system As EventArgs has the event data. Adding the item Males to the ComboBox1. Adding the item Females to the ComboBox1. End of the Form1_Load() sub-procedure. Start of a sub-procedure named ComboBox1_SelectedIndexChanged(). This will be invoked when an item is selected on the first combobox. The sender As Object references the object that raised the event while the e As EventArgs has the event data. Make ComboBox2 empty, clear all items from it. Creating a condition. Checking for whether the selected item on ComboBox1 is Males. Add the item Nicholas to the ComboBox2 when the above condition is true, that is, item selected on ComboBox1 is Male. Add the item John to the ComboBox2 when the above condition is true, that is, item selected on ComboBox1 is Males. Creating a condition. Checking for whether the selected item on ComboBox1 is Females. Add the item Alice to the ComboBox2 when the above condition is true, that is, item selected on ComboBox1 is Females. Add the item Grace to the ComboBox2 when the above condition is true, that is, item selected on ComboBox1 is Females. End of the If block. End of the ComboBox1_SelectedIndexChanged() sub-procedure. End of the class Form1. Summary A ComboBox is created by dragging it from the toolbox and dropping it into the form. It provides us with a way of presenting numerous options to the user. We can set the default item to be selected on the ComboBox when the form is loaded. The SelectedIndexChanged event helps us specify the action to take when a particular item is selected on the combobox. This VB.NET tutorial is a step-by-step guide to learn Visual Basic programming. This free Visual Basic tutorial covers topics like Arrays, Strings, Operators, Switch, Loops, etc. This VB .NET tutorial will help you learn VB.NET basics and advanced concepts. What should I know? This online VB tutorial guide is designed for beginners to learn VB.NET concepts. Refer this link to install Visual Studio. VB.NET Training Syllabus Introduction Advanced Stuff Must Know!