

# ASNA

## Visual RPG for .NET Windows for Smarties

---

*The fast workbook way to learn  
to program Windows with Visual  
RPG for .NET!*

---

Learn the tips and shortcuts for  
developing comprehensive Windows  
applications.

---

Develop an entire Windows application  
by the end of the book—in only four easy  
steps!

---



# Visual RPG for .NET Windows for Smarties

Information in this document is subject to change without notice. Names and data used in examples are fictitious unless otherwise noted.

No component of **Visual RPG for .NET Windows for Smarties** may be reproduced, disassembled, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form without the written permission of ASNA (Amalgamated Software of North America).

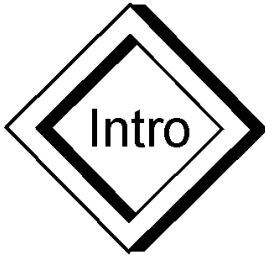
Copyright ©2003 ASNA - Amalgamated Software of North America. All rights reserved.

**Release 7.0 - February 26, 2004**

# CONTENTS

<b>Introduction to Visual RPG for .NET Windows for Smarties.....</b>	<b>1</b>
Using the Tutorial .....	2
Source Files.....	2
Manual Conventions .....	2
<b>Step 1 Installing Necessary Components .....</b>	<b>5</b>
Installing AVR for .NET.....	6
Installation for New Users .....	6
Licensing Visual RPG for .NET IDE.....	7
<b>Step 2: Creating the Windows Form.....</b>	<b>9</b>
Creating the Windows Project and Forms .....	10
Examining Windows Form Structure.....	11
Creating the Windows Form – (Adding Controls and Text).....	12
Running the Windows Application.....	23
Step 2 Summary .....	25
<b>Step 3: Responding to Events .....</b>	<b>27</b>
Adding Code to Form1.vr .....	28
Step 3 Summary .....	32
<b>Step 4: Database Access.....</b>	<b>33</b>
Adding Code to Form1 – Accessing a Database File .....	34
Declaring Database and File Objects .....	34
Connecting to a Database.....	36
Opening a File.....	36
Retrieving Customer Information (btnSearch_Click).....	36
Closing a Database and File (btnExit_Click).....	37
Step 4 Summary .....	38
<b>Appendix A: Visual Studio .NET 101 .....</b>	<b>39</b>
Forms Designer .....	39
Solution Explorer .....	39
Properties .....	40
Toolbox .....	41
Options.....	42
General.....	43
Fonts and Colors .....	43
<b>Index .....</b>	<b>45</b>





# Introduction to Visual RPG for .NET Windows for Smarties

This tutorial will guide you through the steps necessary to write a basic windows application using Visual RPG for .NET.

After completing this tutorial, you will have a working windows application written in AVR that will allow you to connect to a database, retrieve a customer file record, and display the customer details.

---

**At the end of this tutorial, the windows application will look like the following:**

A screenshot of a Windows application window titled "Form1". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area has a light beige background and is titled "Customer Inquiry". It contains a text input field labeled "Enter Customer Number:" followed by a search button and an exit button. Below these are several input fields for customer details: Name, Address, City, State, Zip, Post Code, Phone, and Fax, each with a corresponding label to its left.

## Getting Started

This tutorial does not assume that you have any prior experience with Visual RPG or with creating Windows applications. However, due to the extent of information for Visual Studio .NET and Visual RPG for .NET, you will still want to refer to the on-line documentation for further assistance.

## Using the Tutorial

This tutorial is designed as a “self-learning” tool and includes 4 steps, or main chapters. Each chapter is a “step” in which a portion of the application is created. Each step will build upon the previous step, so it is recommended that you start at Step 1 and progressively work towards the end. Each step should take you approximately an hour or less. The approximate time to complete each step is included at the beginning of each chapter.

### **At the beginning of each chapter, you will find:**

- A summary of the topics that will be covered in that step.
- The approximate time to complete that step.
- What the application will look like at the completion of that step.

### **At the end of each chapter, you will find:**

- A reference section of each task performed in that step, including how to perform that task, and the button or shortcut key that is used to perform that task (if any).

### **At the end of this tutorial, you will find:**


- **Appendix A** – Visual Studio .NET Tools. This appendix describes some of the Windows, or views you have in Visual Studio .NET, and information on getting around in Visual Studio .NET.

## Source Files

There are no source files that accompany this tutorial (as compared to other ASNA Smarties tutorials). Since there is such a small amount of code to enter, it is not necessary to include the completed steps for you to compare your forms or code to.

## Manual Conventions

The following is a list of conventions that are used throughout this tutorial representing particular functions.

- A  beside a headline indicates a step-by-step process that you are to perform.
- Numbered lists (1, 2, 3, and so on) indicate steps that you should follow.
- Menu options, buttons, controls and keys that you are to select, along with

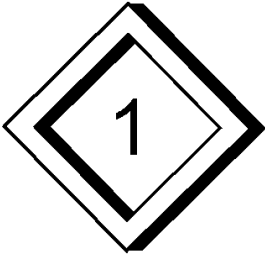
words that you are to type will appear in **boldface**.

- There are several icons that will appear in the left margin, next to the instructions. These icons are a visual representation of a task to perform, a button to select, text to enter, or a Windows form address to enter, such as shown below:



Indicates that you are to select a particular control from the ToolBox (the icon will vary depending upon the control to select).

**This Form Intentionally Left Blank**



## **Step 1**

# **Installing Necessary Components**

---

### **What you will learn in Step 1:**

- Installing Visual RPG for .NET for previous and new users.
- Licensing Visual RPG for .NET.

### **Approximate Time to Complete Step 1:**

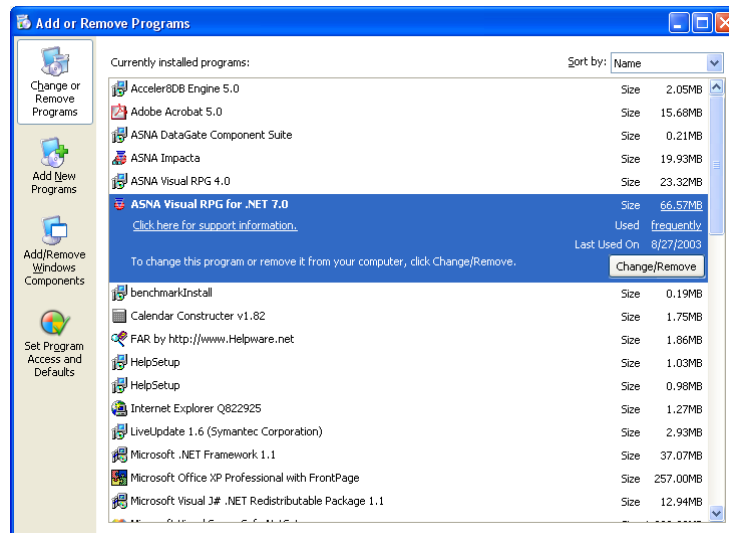
Will vary depending upon the components to install, but could take up to an hour.

## Installing AVR for .NET

### ✓ Installing AVR for .NET for previous Users

1. If this is **NOT** your first installation of AVR for .NET, you must first uninstall the previous version by going to the **Start Menu - Control Panel - Add/Remove Programs**.

Your computer will compile a list of programs currently installed on your machine in a window that looks something like this:



2. Click on **ASNA Visual RPG for .NET 7.0** and select the **Change/Remove** button.
3. In the window that pops up next, select **Remove** and click **Next**.
4. Once the older version of the product is uninstalled, you may continue to the next section and install the current build as if you are a new user.

## Installation for New Users

### ✓ Installing AVR for .NET for New Users

1. If this is **NOT** your first installation of AVR for .NET, you must first uninstall the previous version by going to the **Start Menu - Control Panel - Add/Remove Programs**.

If this is your first time installing AVR.NET (i.e. you do not have any previous version on your machine), then installation is very straightforward.

Simply run the setup file, and accept all the default settings (this equates to basically clicking Next a bunch of times). Once this is complete, you are ready to **license** your copy of AVR for .NET as per the following steps.

## Licensing Visual RPG for .NET IDE

To complete this tutorial, you must have a valid license of Visual RPG. Obtain your license key from ASNA. Enter the license key using ASNA's registration and licensing tool called **ASNA Registration Assistant**.

Follow the steps below to license the Visual RPG for .NET IDE.

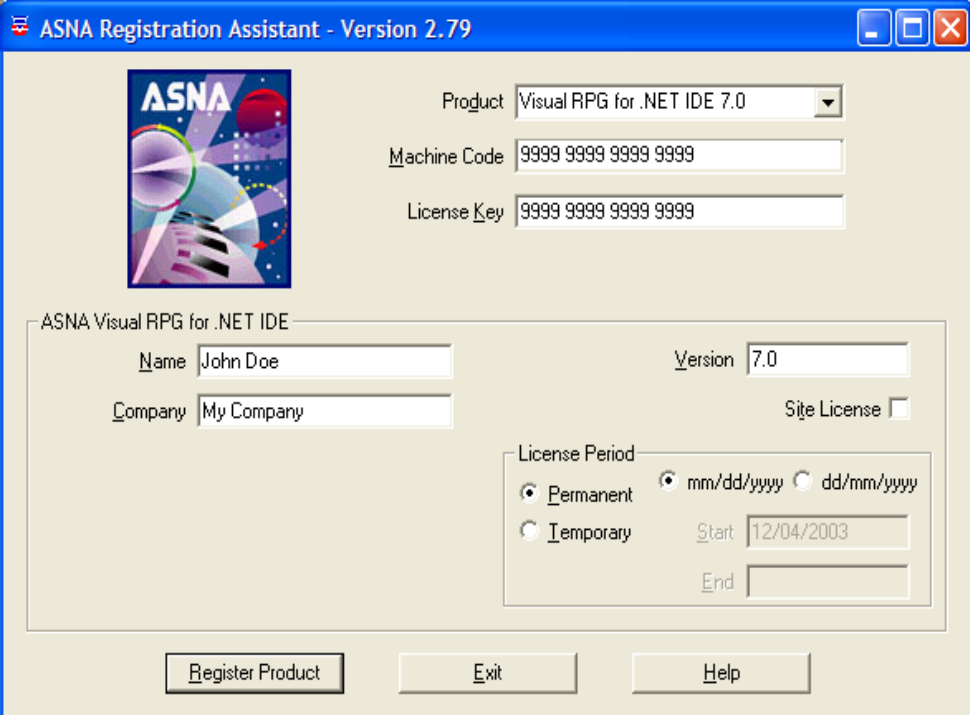
If you have never licensed the Visual RPG for .NET product suite, then you must follow these steps in order to use the product. ***Bear in mind that many of the beta builds did not require licenses to run, so just because you have been using a beta version does not mean that you are automatically registered.***

### To License Visual RPG for .NET IDE 7.0

1. Contact ASNA to obtain a License Key.
2. Once you are given the License Key, enter the License Key within ASNA Registration Assistant. Start ASNA Registration Assistant by selecting **Start - ASNA Product Suite - ASNA Registration Assistant**.

*You can also select **asnalic.exe** from the `\Program Files\Common Files\Asna Shared` folder.*

3. The Registration Assistant dialog will display. You must first select the Product that you are licensing. Select **Visual RPG for .NET IDE 7.0**.
4. Enter the **License Key** that you received from ASNA, along with the other pertinent information, such as Name, Company, and License Period.
5. Select the **Register Product** button to accept the information, the **Exit** button to close Registration Assistant, or the **Help** button to get help on ASNA Registration Assistant.



ASNA Registration Assistant - Version 2.79

Product: Visual RPG for .NET IDE 7.0

Machine Code: 9999 9999 9999 9999

License Key: 9999 9999 9999 9999

ASNA Visual RPG for .NET IDE

Name: John Doe      Version: 7.0

Company: My Company      Site License:

License Period:

Permanent       mm/dd/yyyy       dd/mm/yyyy

Temporary      Start: 12/04/2003

End: \_\_\_\_\_

Register Product      Exit      Help

**Congratulations!**

---

**You should have all of your Windows components installed and licensed, so you are now ready to begin creating your Windows forms in the next step.**



## Step 2: Creating the Windows Form

---

### What you will learn in Step 2:

- How to create a windows project and windows form in Visual Studio .NET.
- How to view the ToolBox and add input elements to a form.
- How to align a group of controls.
- How to run your windows application.

### Approximate Time to Complete Step 2:

1 hour.

### What the Windows Form will look like after completing Step 2:

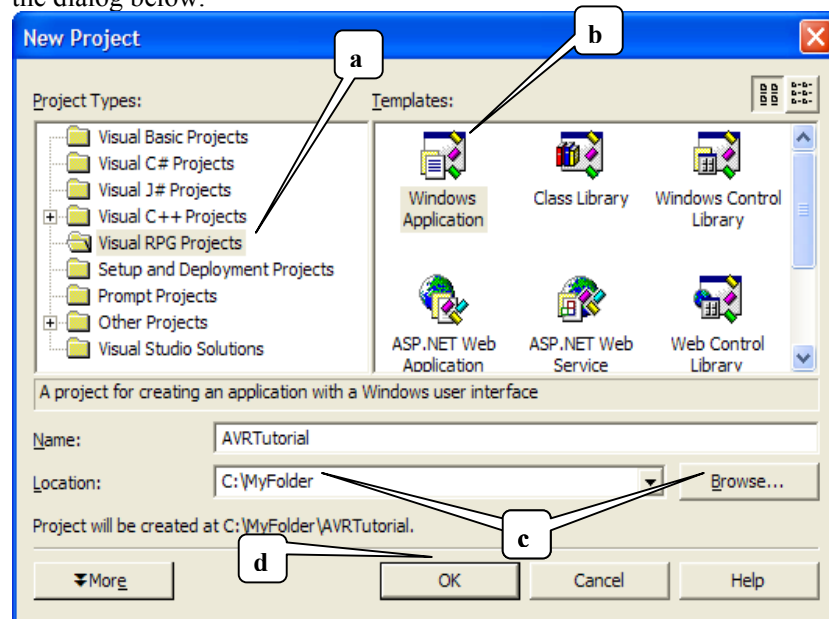
The screenshot shows a Windows application window titled "Form1". The form has a blue title bar with standard minimize, maximize, and close buttons. The main content area has a light gray background with a dotted grid. At the top, the text "Customer Inquiry" is centered. Below it, the text "Enter Customer Number:" is followed by a text input field. Underneath are two buttons: "Search" and "Exit". The lower portion of the form contains a vertical list of labels and text input fields: "Name:", "Address:", "City:", "State:", "Zip:", "Post Code:", "Phone:", and "Fax:".

## Creating the Windows Project and Forms

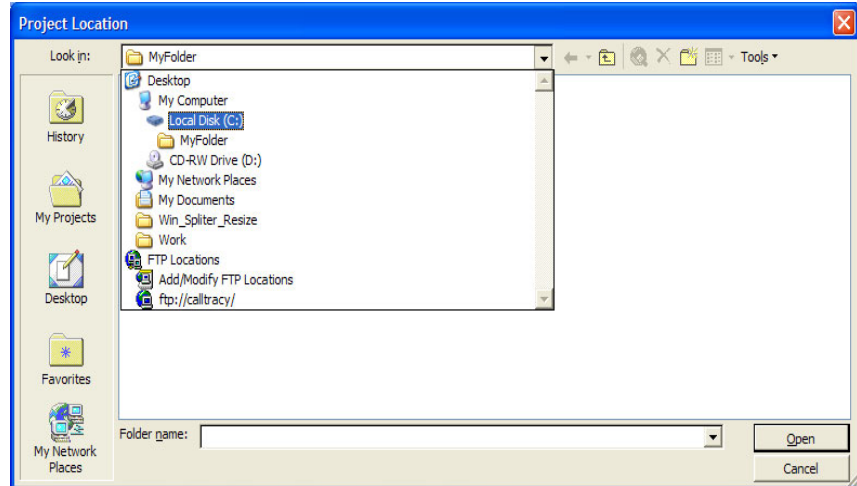
First, we will create a Windows application (which automatically gives you a Windows form).

### ✓ To Create a Windows Project and Windows Form

1. Open **Visual Studio .NET**.
2. On the **File** menu, point to **New**, then click **Project**, or **Ctrl+Shift+N**.
3. In the **New Project** dialog box, perform the following steps while referring to the dialog below.



- a. In the **Project Types** pane on the left, choose **Visual RPG Projects**.
- b. In the **Templates** pane on the right, choose **Windows Application**.
- c. Choose a location on your computer's hard drive to save the new project by clicking on the **Browse** button. A dialog box similar to the one below will appear. Navigate to the location of your choice.

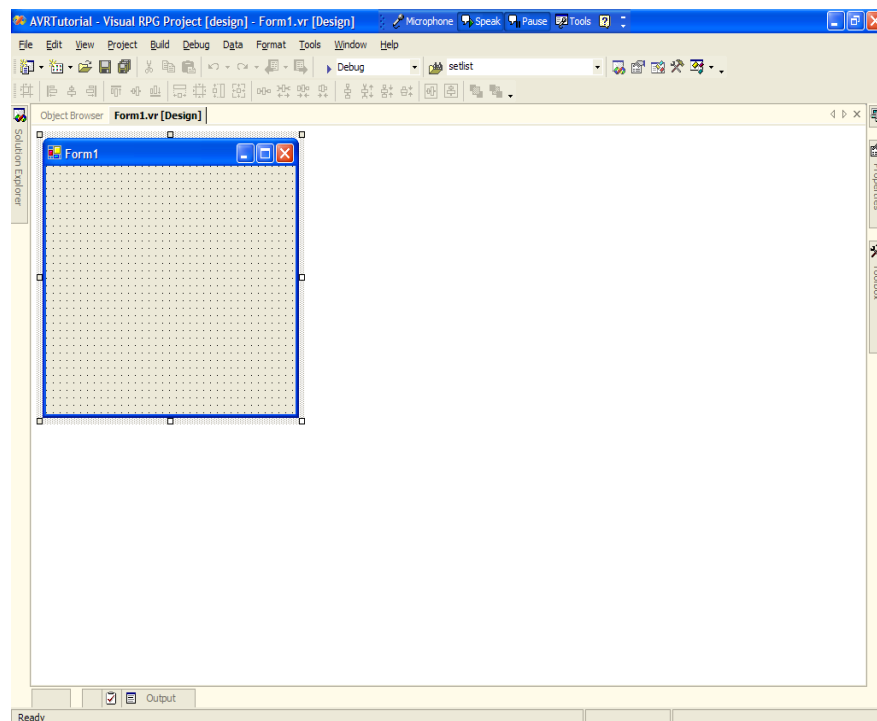


Once selected, the **Location** box will fill in with your chosen location.

- d. Click **OK**. When you click **OK**, the IDE will automatically perform the necessary steps to set up your *AvrTutorial* on your machine. The *AvrTutorial* folder will be created at the root level of the Windows folder you selected in the previous step.

## Examining Windows Form Structure

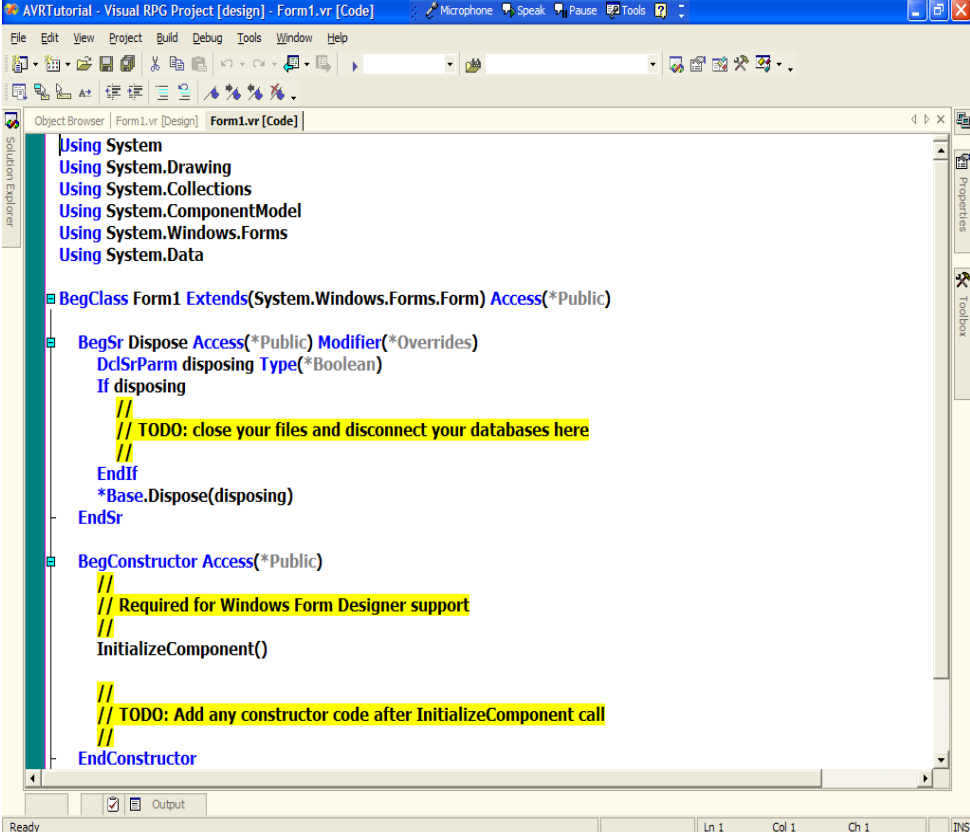
The Windows Form Designer is opened with a file called **Form1.vr [Design]**. This is the form for which you will begin creating the user interface for the application. Notice the word **[Design]** after **Form1.vr**. This denotes that you are viewing the form in design mode.



## 12 Visual RPG for .NET Windows for Smarties

Right-mouse-click on the form and choose **View Code** from the context menu that appears. A new window will appear displaying the code form for the form.

Notice the new tab next to the word **[Code]** after **Form1.vr**. This denotes that you are viewing the code for the form.



```
AVRTutorial - Visual RPG Project [design] - Form1.vr [Code]
File Edit View Project Build Debug Tools Window Help
Object Browser Form1.vr [Design] Form1.vr [Code]
Solution Explorer Properties Toolbox
Using System
Using System.Drawing
Using System.Collections
Using System.ComponentModel
Using System.Windows.Forms
Using System.Data

BegClass Form1 Extends(System.Windows.Forms.Form) Access(*Public)
  BegSr Dispose Access(*Public) Modifier(*Overrides)
    DclSrParm disposing Type(*Boolean)
    If disposing
      // TODO: close your files and disconnect your databases here
      //
    EndIf
    *Base.Dispose(disposing)
  EndSr

  BegConstructor Access(*Public)
    // Required for Windows Form Designer support
    InitializeComponent()
    // TODO: Add any constructor code after InitializeComponent call
  EndConstructor
```

## Creating the Windows Form – (Adding Controls and Text)

You are now ready to begin creating your first Windows form by adding controls to it.

This form will be where the user enters the customer number and views the customer's information retrieved from the file in the database. To begin, let's place a TextBox onto the Windows form where the user will enter the customer number.

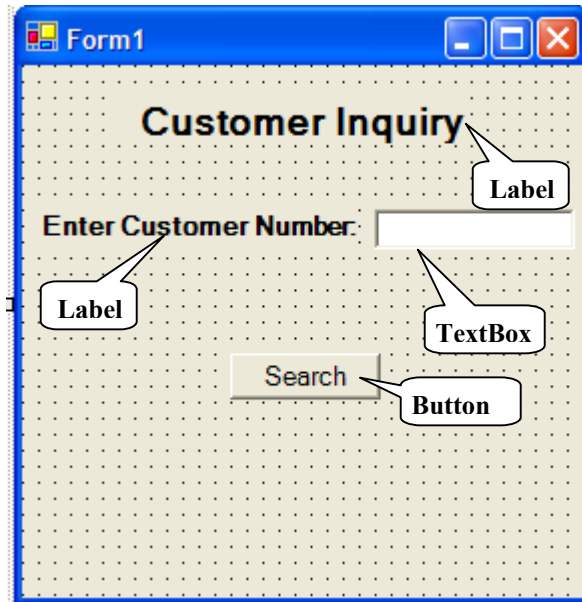
The controls available for use on the Windows form are contained within the **Toolbox** in Visual Studio .NET. The Toolbox displays a variety of items for use in Visual RPG projects. The **tabs** and **items** available from the Toolbox change, depending upon the designer or editor currently in use. The items available can include .NET components, COM components, code fragments, and text.


The Toolbox always displays two tabs, a General tab and a Clipboard Ring tab. As you open an editor or designer, other tabs and tools are displayed. You also can add your own custom tabs and tools to the Toolbox. For more information,

see [Using the Toolbox](#) and [Managing Tabs and Items in the Toolbox](#) in the Visual Studio help.

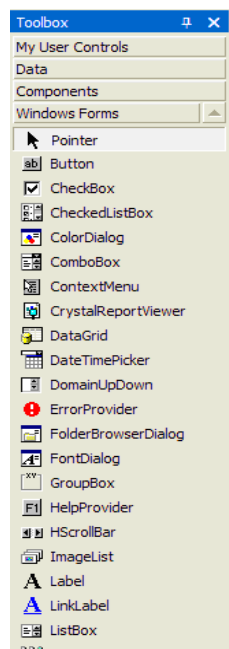
### Let's Create the Form

The form that you will create will look similar to the following:



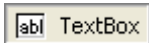
1. Display the **Toolbox** containing the controls by selecting **View -  Toolbox**, or by pressing the **Ctrl + Alt + X** keys.

*Notice that the controls available for Windows forms are displayed under the **Windows Forms** tab.*



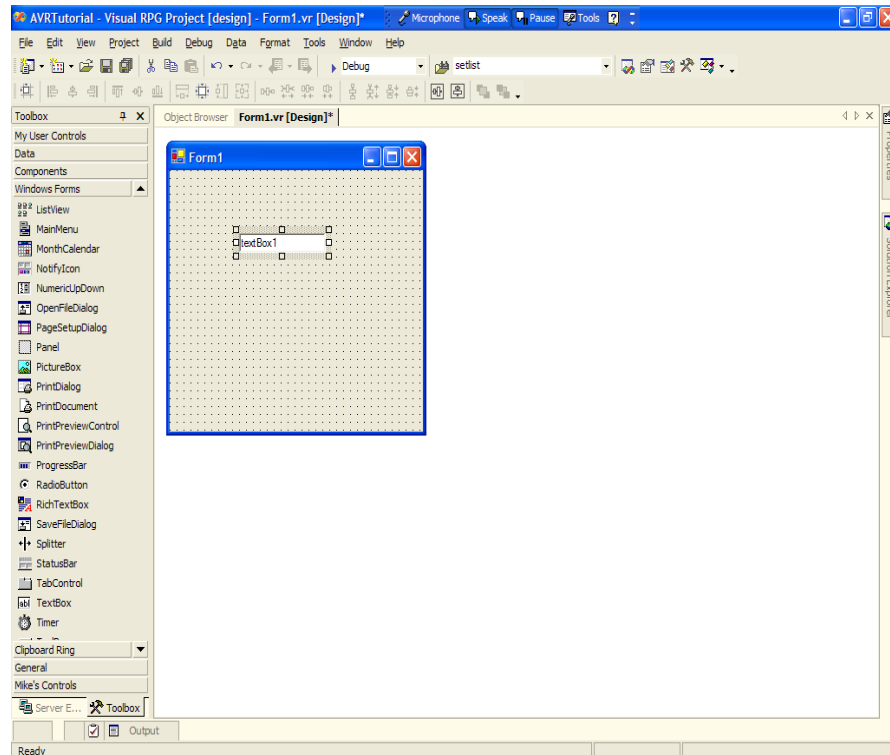
## 14 Visual RPG for .NET Windows for Smarties

You may want to click on the other tabs, such as **Data** or **Components** (just to name a few) and notice that the controls available will vary.



2. Select the **TextBox** control (in the Windows Forms tab) and drag and drop onto the Windows form.

Your Windows form should look like this:



You can click on the control to **move** it, or you can **resize** the control by grabbing one of the square "handles" around it with the mouse pointer.

3. Next, let's change the default name of the TextBox control (TextBox1) to a meaningful name we can recognize in code by changing its **Properties**.

The **Properties** window allows you to view and change the design-time properties and events of selected objects that are located in editors and designers. You can also use the **Properties** window to edit and view file, project, and solution properties. **Properties Window** is available from the **View** menu.

The **Properties** window displays different types of editing fields, depending on the needs of a particular property. These edit fields include edit boxes, drop-down lists, and links to custom editor dialog boxes. Properties shown in gray are read-only. See [Properties Window](#) in the Visual Studio help for more information.

- a. If your **Properties** window isn't already visible, press **F4** to display it.

- b. Ensure that the text box you just placed on the form is selected, and then find the field in the Properties window called "**Name**". *If Properties are presented alphabetically, Name will appear near the top of the list with the default name TextBox1.*
- c. Highlight the name **TextBox1** and replace it with "**txtCustNo**".
- d. Also find the **Text** property and delete the default **TextBox1** there. We don't want a default value for the text, as the user will be entering data.

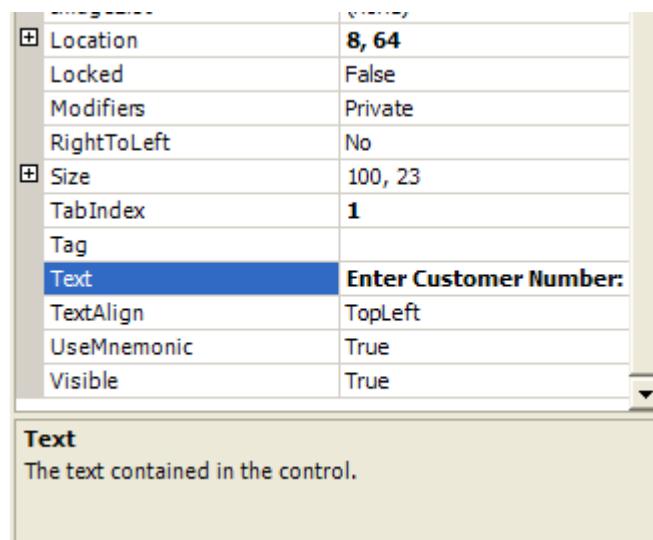
---

*It is suggested that all 'like' controls start with the same prefix (e.g. all TextBox names begin with "txt", and all button names begin with "btn"). This makes it easy to see what type of control you are referencing later on when you are coding.*

---

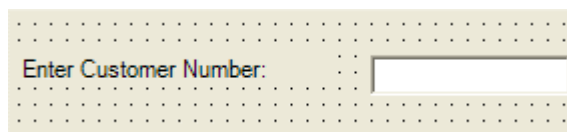
**A** Label

- 4. Let's add a **Label** control next to the TextBox to indicate what this TextBox is asking for.
  - a. Click and drag the **Label** control in the Toolbox onto the area you want the Label control to be on the form.
  - b. Change its **Text** property to "**Enter Customer Number:**"

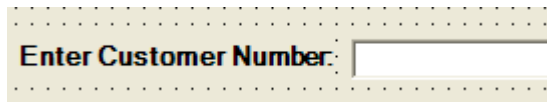


*Changing the name of the label is not usually necessary since you will probably never need to reference this control in the code.*

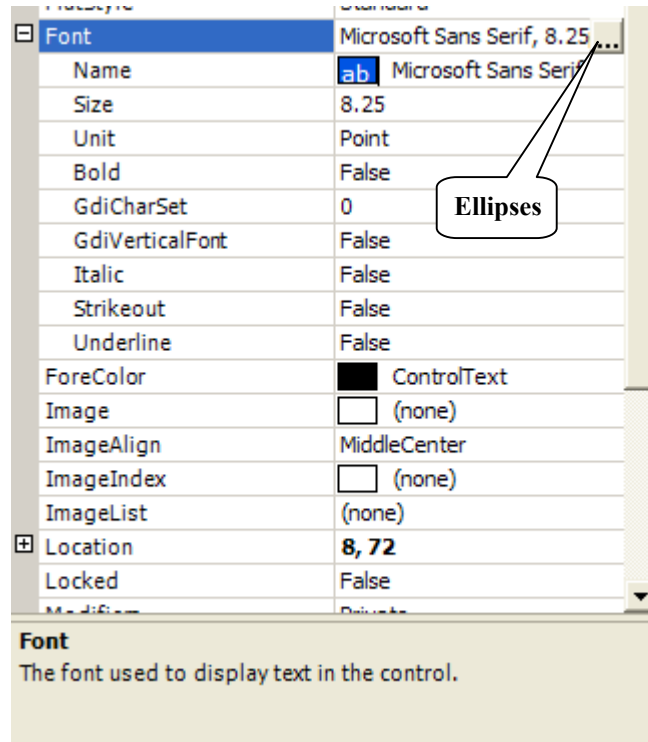
**Your Label control may now look something like the following:**



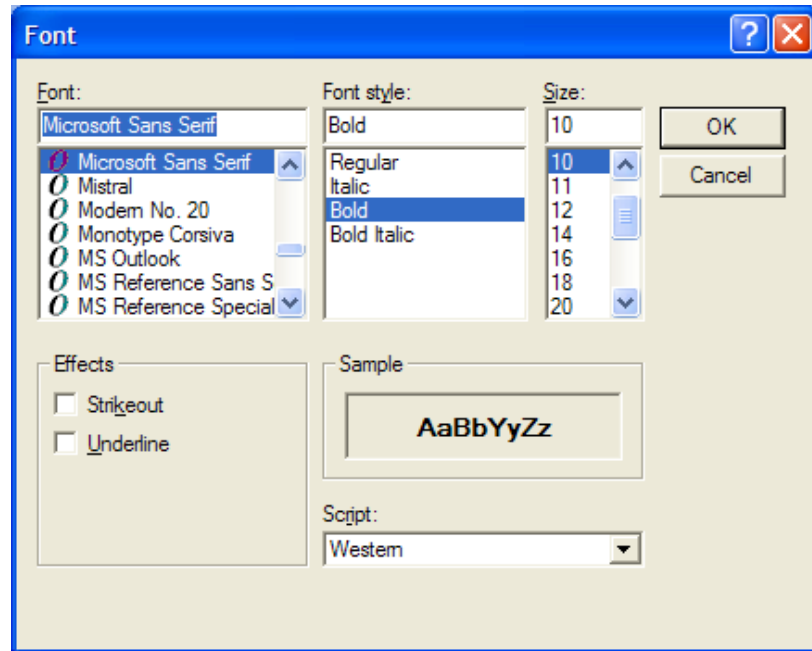
***and we want it to look like this:***



- 5. Let's change our **font** and **size** of our **Label** control.
  - a. Double-click the + sign to the left of the **Font** property in the Properties Window to display the font properties, as shown below.



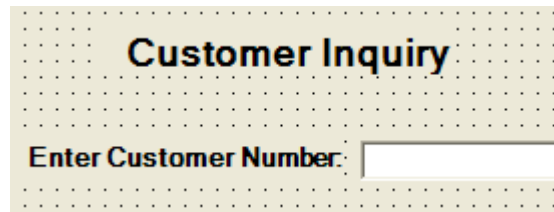
- b. You can change each individual property of the font separately or you can click on the ellipses button to bring up the font dialog box as shown below.



c. For this example, we have chosen not to change the default font. We will however change the **Font style** to **Bold** and the **Size** to 10.

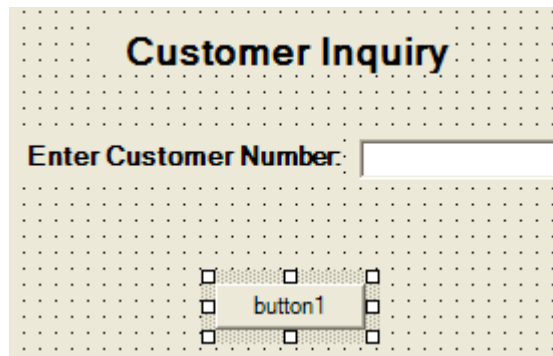
 Label

6. While we're using the **Label** control, let's go ahead and give this form a **Title** by adding another **Label** control.
  - a. Click and drag the **Label** control in the ToolBox onto the top area of the form.
  - b. Change its **Text** property to "**Customer Inquiry**".
  - c. Click the + sign to the left of the **Font** property in the Properties Window, click on the ellipses and change the **Size** property to 14 and the **Font Style** to **Bold**.
  - d. Position the label to the desired location at the **Top** center of the form, similar to the following.



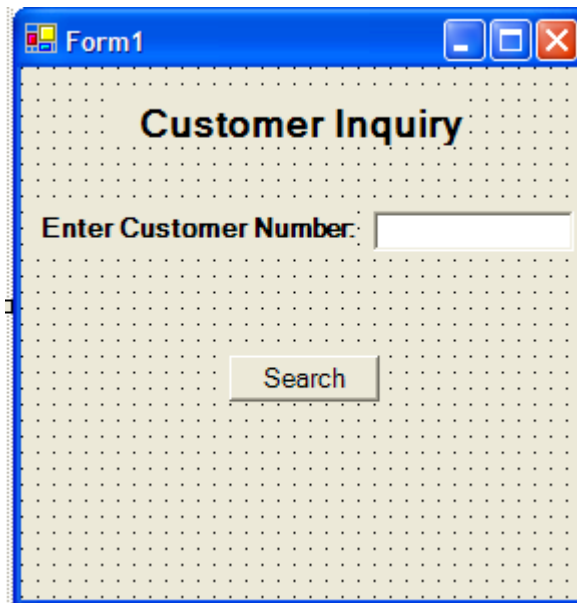
 Button

7. Next, we need to add a **Button** the user will click to initiate the customer search.
  - a. Click and drag the **Button** control onto the form underneath the **Enter Customer Number** label (or anywhere you desire).



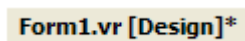
- b. Change its **Name** property from Button1 to “**btnSearch**”.
- c. Change the **Text** property from Button to “**Search**”.
- d. Change the **Font Size** property to 10 (if desired).

**At this point your form should now look similar to the following:**



- 8. We will be adding more elements to the form, but before we go any further, let's save the Windows form by selecting **File – Save Form1.vr**, or select **Ctrl+S**.

*You can tell whether a form needs to be saved by looking at the Form name in a Tab above the form and seeing if there is an asterisk (\*) after the form name, as shown below:*



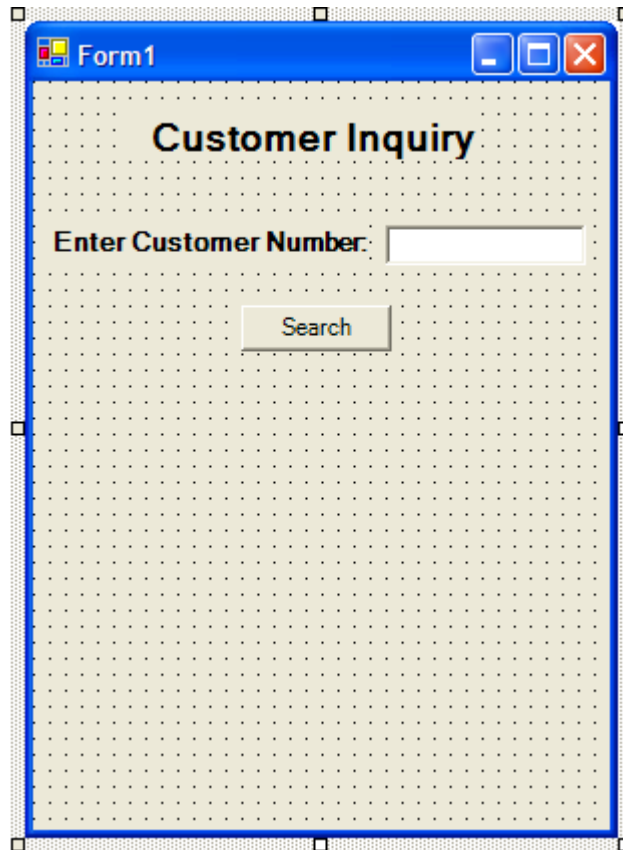
The asterisk indicates that the form needs to be saved, or that a change(s) have been made to the form since it has been last saved.

**Form1.vr [Design]**

When you Save the form, notice that the “asterisk” disappears as shown.

**Let’s continue building our form.**

At this point, we need to increase the size of the form so we will be able to see the customer’s information. To do so, grab the square “handle” on the bottom center of the form with the mouse pointer and drag the form down. Your form should look like the following.



Follow the steps below to create TextBoxes to hold the customer’s **name**, **address**, **city**, **state**, **postal code**, **phone**, and **fax number**, so the form looks like the following:

Form1

## Customer Inquiry

Enter Customer Number:

Name:

Address:

City:





State:


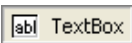
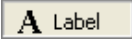

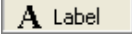
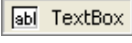

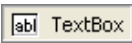
Zip:

Post Code:

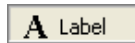
Phone:

Fax:

-  Label
- Name*
1. Let's add a Label for **Name**.
    - a. Click and drag the **Label** control in the ToolBox onto the form.
    - b. Change its **Text** property from **Label** to "**Name:**".
-  TextBox
- Name*
2. Let's add a Textbox for **Name**.
    - a. Click and drag the **TextBox** control in the ToolBox onto the form.
    - b. Change its **Name** property from **TextBox1** to "**txtName**".
    - c. Clear its **Text** property.
-  Label
- Address*
3. Let's add a Label for **Address**.
    - a. Click and drag the **Label** control in the ToolBox onto the form directly under Name Label.
    - b. Change its **Text** property from **Label** to "**Address:**".
-  TextBox
- Address*
4. Let's add a Textbox for **Address**.
    - a. Click and drag the **TextBox** control in the ToolBox onto the form directly under the Name Textbox.
    - b. Change its **Name** property from **TextBox1** to "**txtAddress**".

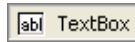
- c. Clear its **Text** property.
-  **A** Label 5. Let's add a Label for **City**.  
*City*
- a. Click and drag the **Label** control in the ToolBox onto the form directly under Address Label.
- b. Change its **Text** property from **Label** to "**City:**".
-  **abl** TextBox 6. Let's add a Textbox for **Address**.  
*City*
- a. Click and drag the **TextBox** control in the ToolBox onto the form directly under the Address Textbox.
- b. Change its **Name** property from **TextBox1** to "**txtCity**".
- c. Clear its **Text** property.
-  **A** Label 7. Let's add a Label for **City**.  
*State*
- b. Click and drag the **Label** control in the ToolBox onto the form directly under City Label.
- c. Change its **Text** property from **Label** to "**State:**".
-  **abl** TextBox 8. Let's add a Textbox for **Address**.  
*State*
- a. Click and drag the **TextBox** control in the ToolBox onto the form directly under the City Textbox.
- b. Change its **Name** property from **TextBox1** to "**txtState**".
- c. Clear its **Text** property.
-  **A** Label 9. Let's add a Label for **Postal Code**.  
*Postal Code*
- a. Click and drag the **Label** control in the ToolBox onto the form directly under State Label.
- b. Change its **Text** property from **Label** to "**Postal Code:**".
-  **abl** TextBox 10. Let's add a Textbox for **Postal Code**.  
*Postal Code*
- a. Click and drag the **TextBox** control in the ToolBox onto the form directly under the State Textbox.
- b. Change its **Name** property from **TextBox1** to "**txtPostCode**".
- c. Clear its **Text** property.
-  **A** Label 11. Let's add a Label for **Phone**.  
*Phone*
- a. Click and drag the **Label** control in the ToolBox onto the form directly under State Label.
- b. Change its **Text** property from **Label** to "**Phone:**".
-  **abl** TextBox 11. Let's add a Textbox for **Phone**.  
*Phone*
- a. Click and drag the **TextBox** control in the ToolBox onto the form directly under the State Textbox.

- b. Change its **Name** property from **TextBox1** to “**txtPhone**”.
- c. Clear its **Text** property.



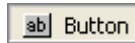
*Fax*

- 12. Let’s add a Label for **Fax**.
- c. Click and drag the **Label** control in the ToolBox onto the form directly under Phone Label.
- d. Change its **Text** property from **Label** to “**Fax:**”.



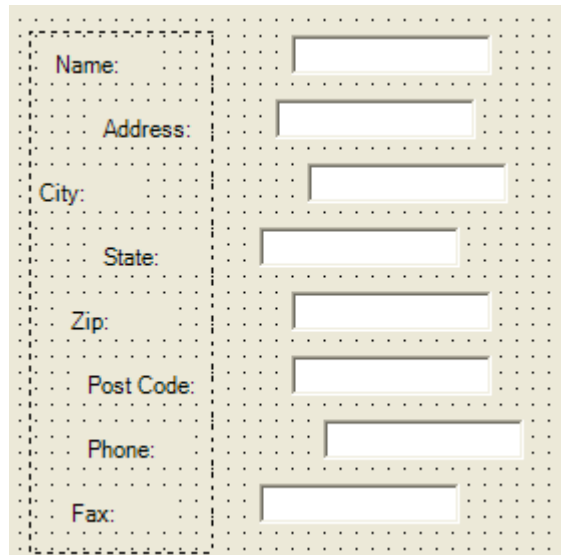
*Fax*

- 13. Let’s add a Textbox for **Fax**.
- a. Click and drag the **TextBox** control in the ToolBox onto the form directly under the Phone Textbox.
- b. Change its **Name** property from **TextBox1** to “**txtFax**”.
- c. Clear its **Text** property.



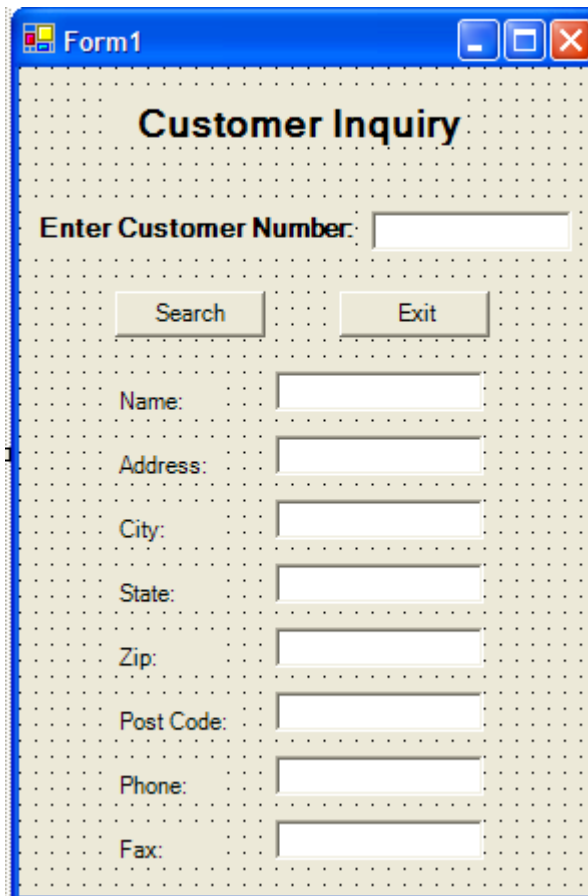
Let’s finally add a Button to allow exiting the application

- a. Click and drag a **Button** control in the ToolBox onto the form and place it next to the **Search** button.
  - b. Change its **Name** property from **button1** to **btnExit**.
  - c. Change its **Text** property to **Exit**.
14. If your form looks like mine right now, we will need to fix the alignment of the Labels and TextBoxes so they appear in straight alignment on the left, and fix the vertical spacing, so there is the same amount of space between each field.
- a. With your mouse, draw a “**Box**” around all of the Label controls, as shown below.



- b. The fields will now have 8 boxes around the fields. Select **Format – Align - Lefts**”. All labels will align to the left to match the alignment of the first field, **Name:**.

- c. Next, “draw” around all of the TextBox fields.
  - d. Select **Format – Align - Lefts**”. All TextBoxes will align to the left to match the alignment of the first field, `txtName`.
  - e. Now, let’s change the vertical alignment, so each field has the same amount of space between the fields.
    - Draw a “box” around the Label controls and select **Format – Vertical Spacing – Make Equal**.
    - Draw a “box” around the TextBox controls and select **Format – Vertical Spacing – Make Equal**.
15. Your completed Windows Form should now look like the following:



The screenshot shows a Windows Form titled "Form1" with a blue title bar and standard window controls. The form has a light gray grid background. At the top, the text "Customer Inquiry" is displayed in a large, bold, black font. Below this, there is a label "Enter Customer Number:" followed by a text box. Underneath are two buttons: "Search" and "Exit". Below these are seven more text boxes, each preceded by a label: "Name:", "Address:", "City:", "State:", "Zip:", "Post Code:", "Phone:", and "Fax:". All text boxes are aligned to the left and have a consistent vertical spacing between them.

## Running the Windows Application





### Let's Run the Windows Application

1. Run your Windows application by selecting **Debug - Start**, or by simply pressing **F5**. (Running the application will “save” the form with the same name).

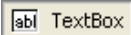
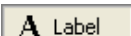
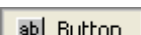
Your code will be compiled and your first form will display on your screen as follows:

The image shows a Windows application window titled "Form1". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The main content area has a light beige background and is titled "Customer Inquiry" in a large, bold, black font. Below the title, there is a text input field with the label "Enter Customer Number:". Underneath this field are two buttons: "Search" and "Exit". Below the buttons are several more text input fields, each with a label to its left: "Name:", "Address:", "City:", "State:", "Zip:", "Post Code:", "Phone:", and "Fax:". All input fields are currently empty.

*Because we have written no code, this form doesn't do anything yet! Adding code will be addressed in the next step.*

2. To stop your Windows application from running, you can simply click the  in upper right hand corner of the form. In Visual Studio, while the project is running, you will also see the pause  and stop  buttons. Clicking on the stop  button will also stop the application from running.

## Step 2 Summary

To	Do This	Button/Keys
Create a Windows application	Select <b>File - New – Project - Visual RPG Projects - Windows Application.</b>	
View the ToolBox	Select <b>View – Toolbox.</b>	<b>Ctrl+Alt+X</b>
Insert a Text Box	Select the <b>TextBox</b> control and drag and drop onto the Windows form.	
Insert a Label	Select the <b>Label</b> control and drag and drop onto the Windows form.	
Insert a Button	Select the <b>Button</b> control and drag and drop onto the Windows form.	
Resize a Control	Click on the control, then click one of the square <b>"handles"</b> with the mouse pointer to resize to the desired dimensions.	
Align a group of controls	Select the appropriate option from the following aligning and sizing options:  <b>Format – Align</b> <b>Format – Make Same Size</b> <b>Format – Horizontal Spacing</b> <b>Format – Vertical Spacing</b>	
Run a Windows Application	Select <b>Debug – Start.</b>	<b>F5</b>
Save a Windows Form	Select <b>File – Save.</b>	<b>Ctrl+S</b>

**This Form Intentionally Left Blank**



## Step 3: Responding to Events

---

### What you will learn in Step 3:

- How to write code to respond to a button being clicked.

### Approximate Time to Complete Step 3:

Half an hour.

### What the code for Form1 will look like after completing Step 3:

#### Form1.vr [Code]

```
BegSr btnSearch_Click Access(*Private) Event(*this.btnSearch.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)

  MsgBox Msg( "Programming with AVR for .NET is fun!" )
EndSr

BegSr btnExit_Click Access(*Private) Event(*this.btnExit.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)

  ExitApp
EndSr
```

## Adding Code to Form1.vr

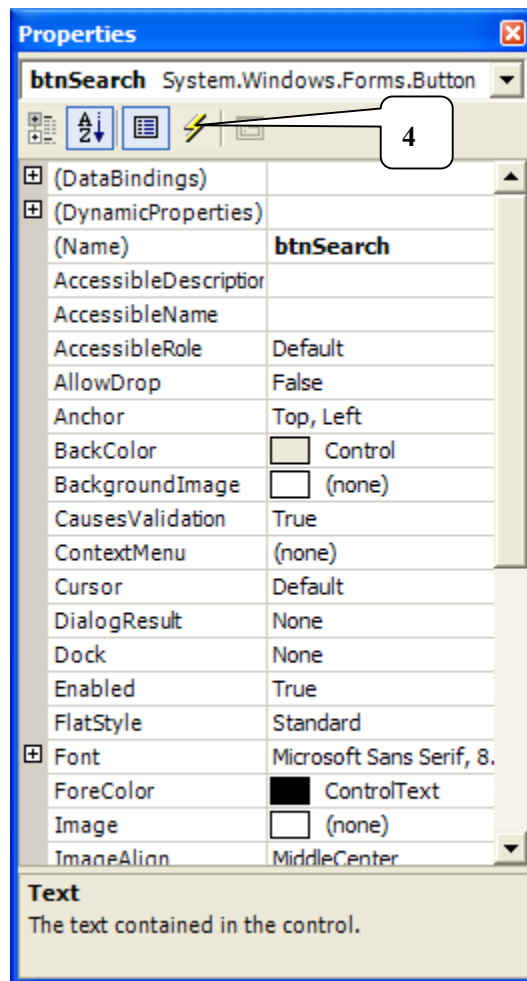
At this point, we will write some code to show how this form will respond when the **Search** and **Exit** buttons are clicked.

### ✔ Let's Add Code to Form1.vr

1. Go to **Form1.vr[Design]** by clicking on the tab.

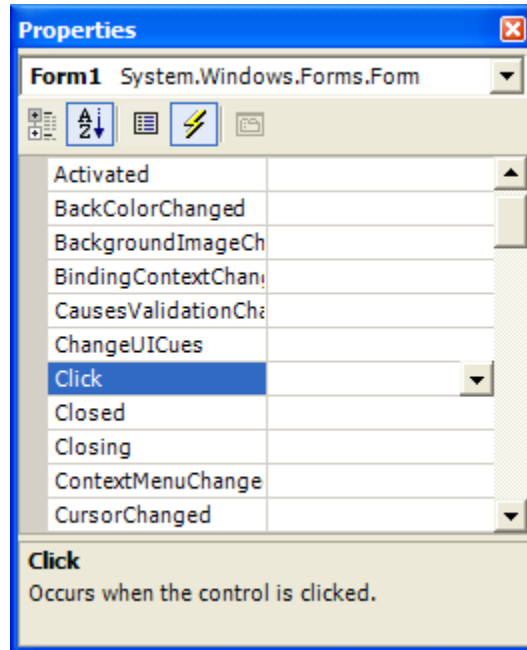
*What we want this form to do for now is to respond to the clicking of the buttons. So first, let's write the code in the **Click** event of **btnSearch**.*

2. Click on the **Search** button "**btnSearch**" on **Form1.vr** to select it.
3. View your **Properties** Window (if it is not currently up, press **F4** to display it). You will see its properties displayed in the Properties window, and it should look like the following:



4. However, we want to access the **Events** of this button, not its Properties, so click on the **yellow lightning bolt** on the top of your Properties window.

This will display all the **events** that can occur on that button. Near the top of the list is the "**Click**" event.



5. Double-click on the **Click** event name and the IDE will automatically take you to the code page for Form1.

*Notice that another **Tab** is displayed at the top with the other tabs displaying **Form1.vr[Code]\*** as displayed below:*

**Form1.vr [Code]\***

*All Visual RPG files will have the .vr extension at the end.*

The code editor automatically generates the event subroutine that handles the click. Your code will now look like this:

```

BegSr Dispose Access(*Public) Modifier(*Overrides)
  DclSrParm disposing Type(*Boolean)
  If disposing
    //
    // TODO: close your files and disconnect your databases here
    //
  EndIf
  *Base.Dispose(disposing)
EndSr

BegConstructor Access(*Public)
  //
  // Required for Windows Form Designer support
  //
  InitializeComponent()

  //
  // TODO: Add any constructor code after InitializeComponent call
  //
EndConstructor

Windows Form Designer generated code

BegSr btnSearch_Click Access(*Private) Event(*this.btnSearch.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)
EndSr

EndClass

```

Your cursor will already be placed at the beginning of your new subroutine, which is named “**btnSearch\_Click**”. Any code placed within this subroutine will be executed whenever a user clicks the **Search** button while running the Windows application.

Let’s add just **one line** of code that will display a windows message box on the screen. We will change this line of code later, but for now, let’s see the form react and show us the message box when the button is clicked.

6. Lets insert a **new line** before the **EndSr**, and enter in the following line:

```
MsgBox Msg( "Programming with AVR for .NET is fun!" )
```

Your **btnSearch\_Click** routine should look like the following:

```

BegSr btnSearch_Click Access(*Private) Event(*this.btnSearch.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)


  MsgBox Msg( "Programming with AVR for .NET is fun!" )
EndSr

```

7. **Run** the project (**F5**), and now when you click on **Search**, the Windows message box should appear, as shown below:

The screenshot shows a Windows application window titled "Form1" with a standard title bar (minimize, maximize, close buttons). The main window has a title "Customer Inquiry" and a form with the following elements:


- Label: "Enter Customer Number:" followed by a text input field.
- Buttons: "Search" and "Exit".
- Label: "Name:" followed by a text input field.
- Modal Dialog Box: A small window with a blue title bar and a close button, containing the text "Programming with AVR for .NET is fun!" and an "OK" button.
- Labels and Text Boxes: "Zip:", "Post Code:", "Phone:", and "Fax:", each followed by a text input field.

8. **Stop** running your Windows application now by clicking the  in the upper right hand corner of the form.
9. Now let's code an event for the **Exit** button. Follow the same steps as you did previously to add the **Click** event in the code editor for the **Exit** button.
10. Insert this line of code in the **Click** event subroutine for the **Exit** button.  
`ExitApp`

ExitApp is an operation code that will end the application.

***Great Job!*** Most of the work is now completed. All that remains is to take the customer number that will be typed into the field and use it to CHAIN to the Customer Master file and present the data to the user.

## Step 3 Summary

To	Do This	Button/Keys
Access the events of a Toolbox control	Click on the <b>Lightning bolt</b> button in the <b>Properties</b> window.	
Write code in the event subroutines for the <b>Search</b> and <b>Exit</b> buttons	<p>The following code caused the Windows message box to appear when the <b>Search</b> button was clicked.</p> <pre>MsgBox Msg( "Programming with AVR for .NET is fun!" )</pre> <p>The following code caused the application to end when the <b>Exit</b> button was clicked.</p> <pre>ExitApp</pre>	



## Step 4: Database Access

### What you will learn in Step 4:

- How to declare database and file objects.
- How to connect to a database and open a file.
- How to retrieve customer information and close a database and file.

### Approximate Time to Complete Step 4:

1 hour +.

### What the Code for Form1 will look like after completing Step 4:

At the end of Step 4, the application code for WindowsForm2 will look like the following:

```

BegSr btnSearch_Click Access(*Private) Event(*this.btnSearch.Click)
  DeclSrParm sender Type(*Object)
  DeclSrParm e Type(System.EventArgs)
  DeclFld CustNo Type( *Packed ) Len( 9,0 )

  CustNo = txtCustNo.Text
  Chain Cust Key( CustNo ) Err( *Extended )
  If ( %Found )
    txtName.Text      = CMName
    txtAddress.Text   = CMAddr1
    txtCity.Text      = CMCity
    txtState.Text     = CMState
    txtPostCode.Text  = CMPostCode
    txtPhone.Text     = CMPhone
    txtFax.Text       = %EditW( CMFax, "0( )& - " )
  Else
    MsgBox Msg( "Customer Not Found" )
  EndIf
EndSr

BegSr btnExit_Click Access(*Private) Event(*this.btnExit.Click)
  DeclSrParm sender Type(*Object)
  DeclSrParm e Type(System.EventArgs)
  Close Cust
  Disconnect AppDB
  ExitApp
EndSr

BegSr Form1_Load Access(*Private) Event(*this.Load)
  DeclSrParm sender Type(*Object)
  DeclSrParm e Type(System.EventArgs)
  Connect AppDB
  Open Cust
EndSr

```

## Adding Code to Form1 – Accessing a Database File

### ✔ Let's Add Code to Form1.vr

Now let's put the program to work accessing the database for the customer data we're searching for by viewing the code for Form1.

1. Go to Form1's code by double-clicking on the form or by using one of the following ways.

---

*Another way to get to the code for any form you are working on is to go to the **Solutions Explorer** window and right click the form and then select **View Code**. Alternatively, you can select **View – Code** from the IDE menu, or press **F7**.*

---

2. Notice the code that is automatically generated by the Windows Designer (Using and DclFld statements, etc). Don't worry about the Using statements for now, but notice the **DclFld** statements. These were generated by the IDE as variables describing all the **objects** you placed onto **Form1**.

```

AVRTutorial - Visual RPG Project [design] - Form1.vr [Code]
File Edit View Project Build Debug Tools Window Help
Object Browser | Form1.vr [Design] | Form1.vr [Code] | Disassembly
Toolbox
Properties

Using System
Using System.Drawing
Using System.Collections
Using System.ComponentModel
Using System.Windows.Forms
Using System.Data

BeginClass Form1 Extends(System.Windows.Forms.Form) Access(*Public)
  DclFld txtCustNo Type(System.Windows.Forms.TextBox) Access(*Private) WithEvents(*Yes)
  DclFld label1 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld label2 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld txtPostCode Type(System.Windows.Forms.TextBox) Access(*Private) WithEvents(*Yes)
  DclFld txtName Type(System.Windows.Forms.TextBox) Access(*Private) WithEvents(*Yes)
  DclFld label4 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld label7 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld txtFax Type(System.Windows.Forms.TextBox) Access(*Private) WithEvents(*Yes)
  DclFld label9 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld txtAddress Type(System.Windows.Forms.TextBox) Access(*Private) WithEvents(*Yes)
  DclFld txtState Type(System.Windows.Forms.TextBox) Access(*Private) WithEvents(*Yes)
  DclFld label10 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld txtZip Type(System.Windows.Forms.TextBox) Access(*Private) WithEvents(*Yes)
  DclFld label6 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld label5 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld txtCity Type(System.Windows.Forms.TextBox) Access(*Private) WithEvents(*Yes)
  DclFld label8 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld label3 Type(System.Windows.Forms.Label) Access(*Private) WithEvents(*Yes)
  DclFld txtPhone Type(System.Windows.Forms.TextBox) Access(*Private) WithEvents(*Yes)
  DclFld btnSearch Type(System.Windows.Forms.Button) Access(*Private) WithEvents(*Yes)

  BeginSr Dispose Access(*Public) Modifier(*Overrides)
    DclSrParm disposing Type(*Boolean)
    If disposing
      //
      // TODO: close your files and disconnect your databases here
    EndIf
  EndSr
EndClass
Output
Ln 263 Col 62 Ch 56
Ready

```

## Declaring Database and File Objects

3. First, let's declare Database and File objects (i.e. "F-Specs"), and code them just **below** that large block of DclFlds as noted in the diagram above.

Enter in the following lines:


```
// Database
DclDB Name( AppDB ) DBName( "DG Net Local" )
// Disk file.
DclDiskFile Name( Cust ) +
                Type( *Input ) +
                File( "*Lib1/CMastNewL1" ) +
                Org( *Indexed ) +
                ImpOpen( *No ) +
                DB( AppDB )
```

The first line of code (**DclDB Name( AppDB ) DBName( "DG Net Local"**) declares a database object that will establish the connection to a Data Source. The **DBName** parameter describes the database name that delineates the database server, user ID, password, etc. In this case, whenever you reference the AppDB variable, you are actually referencing the connection to the “DG Net Local” database. (This database is installed on your local machine, and it operates just as a DB2/400 database).

The **DclDiskFile** lines of code are all part of the same declaration, and they are responsible for creating an “**F-Spec**”.

- Notice the **Name** parameter is **Cust**. This is how you will refer to this file within your code.
- The **File** parameter refers to the CMastNewL1 file in the library list (on the local database, it's in the Examples library).
- The **DB** parameter is the database name you specified previously. This informs the compiler that the format for this file can be found at the location specified within the AppDB database.
- The “+” is the AVR continuation character. All the DclDiskFile parameters can be placed on a single line if you prefer. Multiple lines are used here for readability.

This completes the variable declarations needed to accomplish our task, now let's add some operational code.

4. Move down to the **Form\_Load** event subroutine. If this subroutine is not in your code editor, you will need to add the event, similarly to the way you added the **Click** event for the button. Select the **Form1** design view, and press the **F4** key to bring up the **Properties** window. Click on the lightning bolt  to view the events for the form. Find the **Load** event in the list and double-click on it to add the load event to the code window. The code contained within this subroutine (as the name suggests) is executed **each** time the form is loaded by Windows. Don't worry about the **DclSrParms** in the event. For now, just place all your code **below the DclSrParms and before the EndSr**.

## Connecting to a Database

- Before you can do anything with the file, you must first **connect to the database**, so enter in:

```
Connect AppDB
```

This one line establishes a connection with the database that you will be using throughout the application.

## Opening a File

- Next, you must **open** the file that you will be reading from, so enter the following command:

```
Open Cust
```

## Retrieving Customer Information (btnSearch\_Click)

- The last section of code will do the rest of the work and will go into the click event of the **Search** button:

```
DclFld CustNo Type( *Packed ) Len( 9,0 )

CustNo = txtCustNo.Text

Chain Cust Key( CustNo ) Err( *Extended )

If ( %Found )
    txtName.Text      = CMName
    txtAddress.Text   = CMAddr1
    txtCity.Text      = CMCity
    txtState.Text     = CMState
    txtPostCode.Text  = CMPostCode
    txtPhone.Text     = CMPhone
    txtFax.Text       = %EditW( CMFax, "0( )& - " )
Else
    MsgBox Msg( "Customer Not Found" )
EndIf
```

First, we need to declare a **\*Packed** field that will be used to hold the customer number entered in the customer number text box. Second, we reuse the **MsgBox** line of code by cutting and pasting it into the **Else** portion of our **If** statement, and changing the message displayed.

- If the customer number entered is valid, the customer details are assigned to the Windows TextBoxes.
- Otherwise, the message box appears.

The `btnSearch_Click` event subroutine should now look like the following:

```

BegSr btnSearch_Click Access(*Private) Event(*this.btnSearch.Click)
  DclSrParm sender Type(*Object)
  DclSrParm e Type(System.EventArgs)

  DclFld CustNo Type( *Packed ) Len( 9,0 )

  CustNo = txtCustNo.Text

  Chain Cust Key( CustNo ) Err( *Extended )

  If ( %Found )
    txtName.Text      = CMName
    txtAddress.Text   = CMAddr1
    txtCity.Text      = CMCity
    txtState.Text     = CMState
    txtPostCode.Text  = CMPostCode
    txtPhone.Text     = CMPhone
    txtFax.Text       = %EditW( CMFax, "0( )& - " )
  Else
    MsgBox Msg( "Customer Not Found" )
  EndIf

EndSr

```

### Closing a Database and File (`btnExit_Click`)

Whenever the form is unloaded from the memory, the files and database connections must be closed. For this example we will place the code necessary to do this in the `Click` event for the `Exit` button. (The `Form1_Closed` event is also an optional event to place this code. This event fires when the form is unloaded from memory as well.)

- In the `btnExit_Click` event subroutine, place the following lines of code *before* the `ExitApp` operation code:

```

Close Cust
Disconnect AppDB

```

- Now you can **Run** the entire application by pressing **F5**.

On the form, input the customer number you wish to query.

---

*In the Customer Master example file, all customer numbers end in '00'.  
Valid customer numbers are **100, 200, 300,...100000**.*

---

Your program will **connect** to the database and **open** your file when the application starts. Enter a customer number in the customer number text box and click the **Search** button to **chain** to that record.

**Congratulations,**

You have just completed your first Visual RPG .NET Windows Application!

## Step 4 Summary

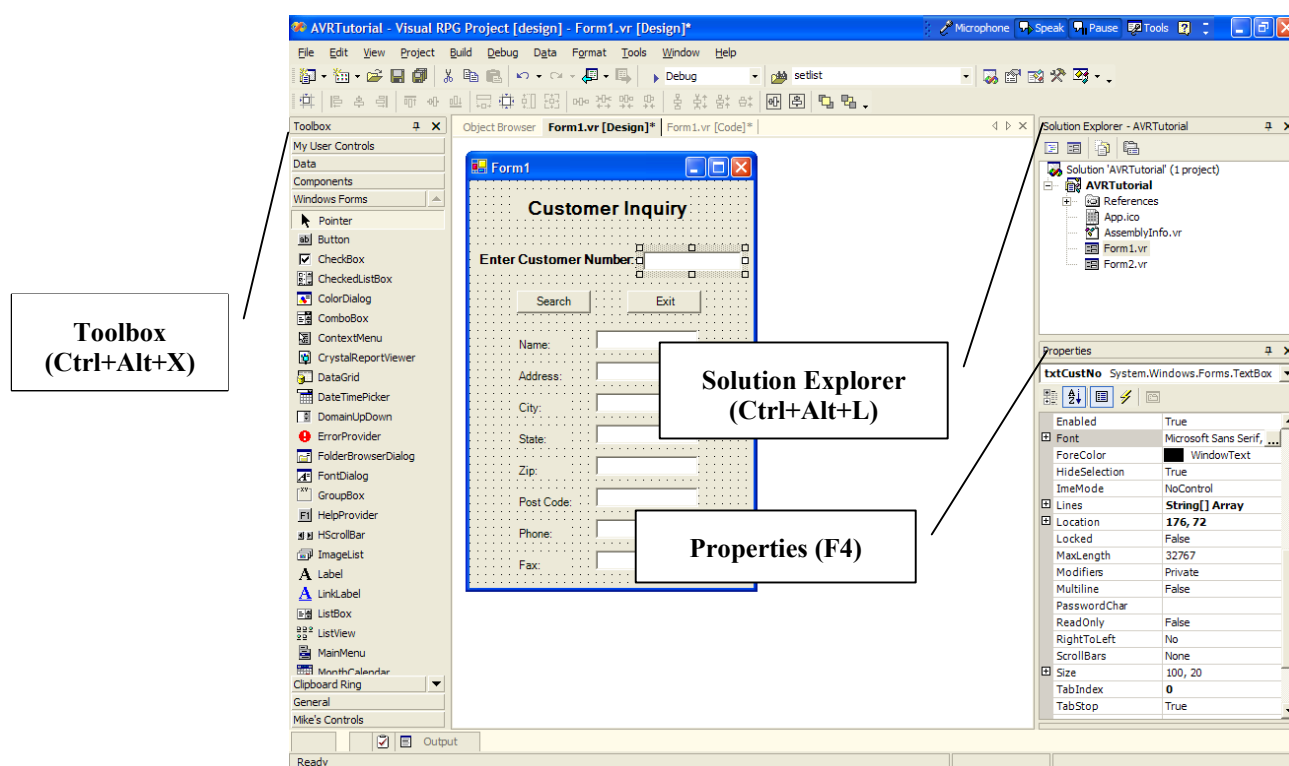
To	Do This	Button/Keys
View Code	Select any of the following: <ul style="list-style-type: none"> <li>• Double-click on the form.</li> <li>• Go to the Solutions Explorer window and right click the form and then select View Code.</li> <li>• Select View – Code from the IDE menu.</li> <li>• Press F7.</li> </ul>	<b>F7</b>
Declare a database object	Sample code: <pre>DclDB Name( AppDB ) DBName( "DG Net Local")</pre>	
Declare a Disk File	Use DclDiskFile	
Connect to a Database	Use the Connect op code followed by the database (same as the Name parameter on the DclDb op code). <pre>Connect &lt;database name&gt;</pre>	
Open a Database file	Code: <pre>Open &lt;file&gt;</pre>	
Disconnecting from a Database	Use the Disconnect op code followed by the database (same as the Name parameter on the DclDb op code). <pre>Disconnect &lt;database name&gt;</pre>	
Close a Database file	Code: <pre>Close &lt;file&gt;</pre>	

## Appendix A: Visual Studio .NET 101

There are a host of built-in tools in Visual Studio .NET that aid in coding your project. This appendix will address the most commonly used tools giving brief descriptions of how to use them to your advantage.

### Forms Designer

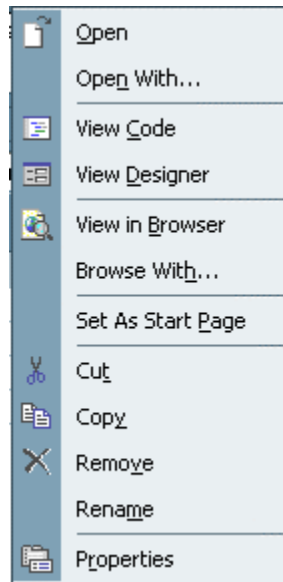
The forms designer is a part of Visual Studio's IDE that allows you the programmer, to design your Windows app or Windows form interface quickly and easily. Here is a typical view of the forms designer with descriptions of the 3 main windows and the shortcut keys to access them:



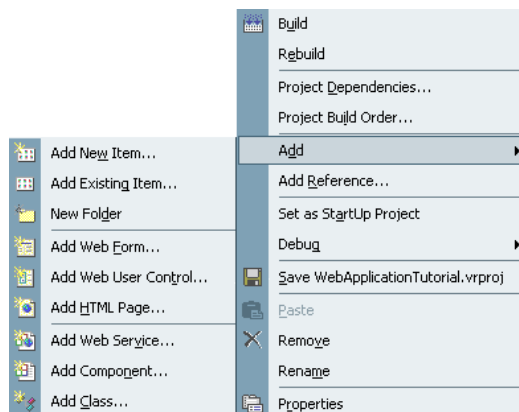
### Solution Explorer

The Solution Explorer is a navigational tool for your solution in much the same way Windows Explorer is a navigational tool for Windows. From the Solution Explorer you can open and edit any file within your project, and any project's files within your solution. These files are presented to you in familiar tree-style

layout, and you can perform a number of operations on the file by right clicking on the file name in the Solution Explorer, and then selecting your choice from the following menu:

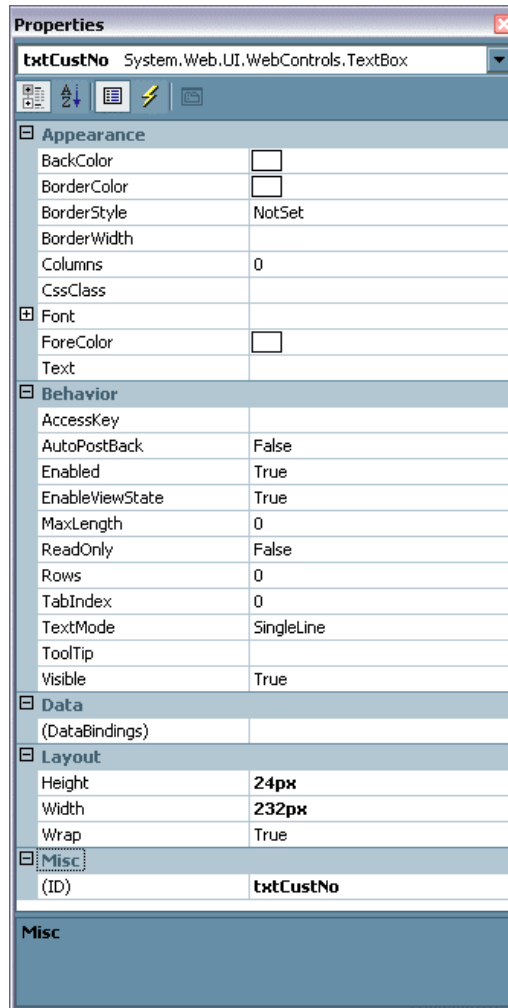


Here you can open the file for editing either the code or the design by selecting either “View Code” or “View Designer”, or you can open the file in a different editor such as notepad by selecting “Open With”. Also, the properties of the file can be viewed by selecting the “Properties” option. By right clicking on the Project itself in the Solution Explorer and selecting “Add”, you can add new files:



## Properties

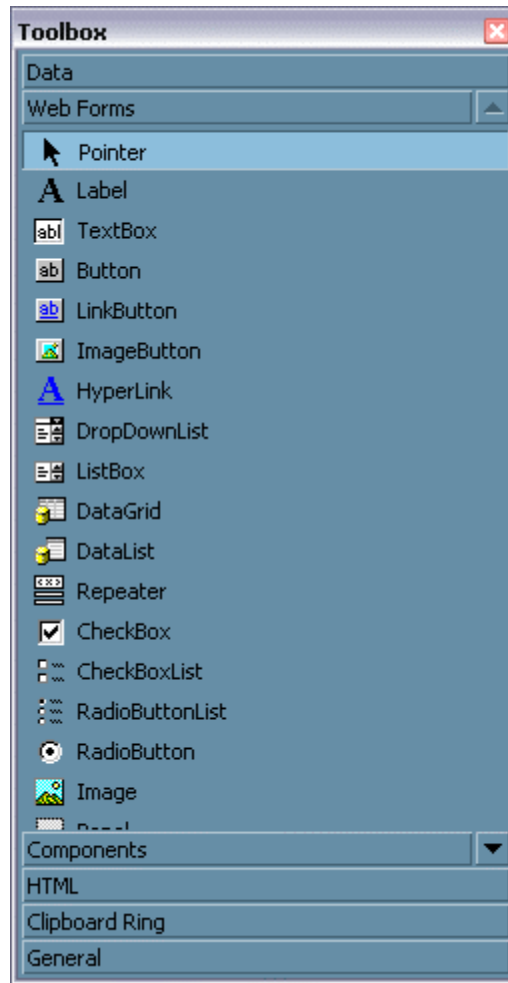
The properties pane provides a means to view and change the properties of controls used on either your Windows or windows forms. When you select a control on your form in design view, the properties window is automatically populated with all the properties for that control. Here is an example of what it may look like for a TextBox control:



Here you can adjust virtually every configurable aspect of the control.

## Toolbox

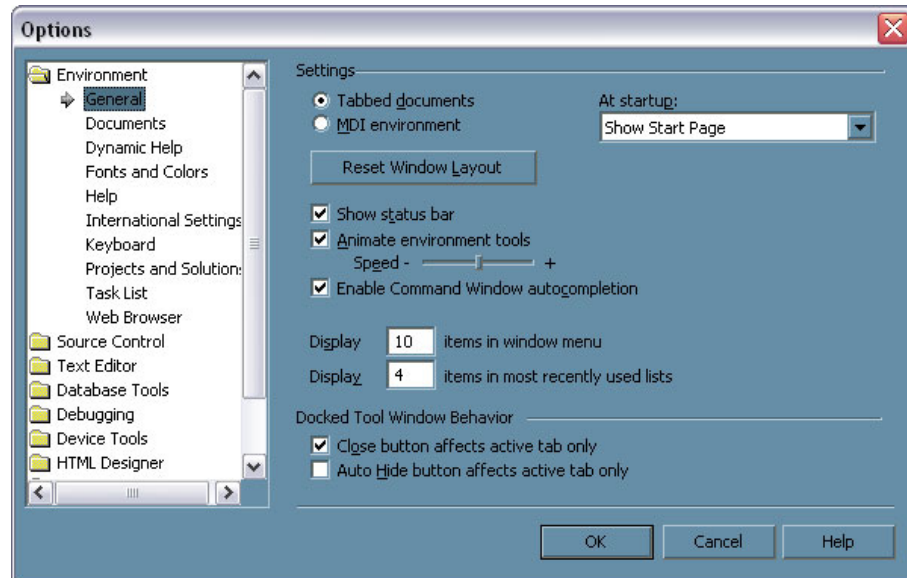
The Toolbox is a vital addition to the forms designer because it holds all the common controls with which you will want to populate your form. The toolbox by default looks like this:



And it is used by simply scrolling through to find the control you wish to use, and then clicking that control to select it. All you have to do then is click and drag the area on your form you want the control to occupy, and it is generated there with all the code automatically created by the compiler to describe it.

## Options

Several other customizations are available to you through the IDE's Tools menu. If you click on **Tools - Options**, the options menu is presented to you:



Most of the changes you will probably want to make will be to the environment itself, customizing it to your liking. Within the environment folder you have many different options, but we will only cover the most commonly used ones here.

## General

Most of these options are self-explanatory. An important one to mention here is the top option button options (Tabbed documents and MDI environment). Here you choose the layout of your files as you work on them. For instance, if you choose Tabbed documents, you will be presented with all of your files that you open in a tabbed format. That is, the environment shows several tabs along the top of the IDE each one providing you access to your files by clicking on the corresponding tab. The other option, MDI environment, will be familiar to all those programmers who have used Visual Studio 6.0 and earlier, or AVR 4.0 and earlier. This format presents all of your files in a window format, with each file receiving its own window held within the IDE.

## Fonts and Colors

This folder is useful for customizing the color-coded text given to you in the text editor. Here you can specify the font type you want used, the size, and the color for virtually every type of special text (i.e. keywords, operators, etc.). For instance, if you want to change the color of all the strings in your projects, simply select Strings from the Display Items Menu, and change the color to that of your liking.

**This Form Intentionally Left Blank**

# Index

- Accessing
  - events, 29
- Adding
  - button control, 18
  - code to Form1, 29
  - code to WindowsForm2, 36
  - label control, 16
  - textbox control, 15
- Aligning
  - controls, 23
- Appendix A
  - fonts and colors, 47
  - forms designer, 43
  - general info, 47
  - options, 46
  - properties, 45
  - solution explorer, 44
  - toolbox, 45
- Application
  - running, 24
- ASNA Registration Assistant, 7
- AVR for .NET
  - installing for new users, 6
  - installing for previous users, 6
- BtnSearch\_Click event
  - coding, 31
- Button control
  - adding, 18
- Changing
  - font size, 17
- Click event
  - coding, 30
- Closing
  - file, 41
- Code
  - adding to Form1, 29
- Coding
  - btnSearch\_Click event, 31
  - click event, 30
  - DclDiskFile, 38
  - Page\_Load, 38, 40
  - Page\_UnLoad, 41
  - WindowsForm2, 36
- Connect
  - coding, 40
- Connecting to
  - database, 40
- Controls
  - aligning, 23
  - button, 18
  - label, 16
  - sizing, 23
  - textbox, 15
- Creating
  - request form, 13
  - Windows application, 10
- Database objects
  - connecting to, 40
  - declaring, 38
  - disconnecting, 41
- DclDiskFile
  - coding, 38
- Declaring
  - database objects, 38
  - file objects, 38
- Events
  - accessing, 29
  - Page\_UnLoad, 41
- F5, 31
  - running application, 24
- File
  - closing, 41
- File objects
  - declaring, 38
  - opening, 40
- Fonts and colors
  - appendix A, 47
- Form1
  - adding code to, 29
- Forms Designer
  - appendix A, 43
- General information
  - appendix A, 47
- Getting started, 2
- How Windows forms are structured, 12
- Installing
  - AVR for .NET for new users, 6
  - AVR for .NET for previous users, 6
- Label control
  - adding, 16
- Manual conventions, 2
- New users
  - installing AVR for .NET, 6
- Opening
  - file, 40
- Options
  - appendix A, 46
- Page\_Load
  - coding, 38, 40
- Page\_UnLoad
  - coding, 41
- Previous users

- installing AVR for .NET, 6
- Properties
  - appendix A, 45
  - viewing, 15
- Redirecting
  - to Form1, 41
- Running
  - the application, 31
  - Windows application, 24
- Running application
  - pressing F5, 24
- Sizing
  - controls, 23
- Solution Explorer
  - appendix A, 44
- Source files, 2
- Step 1
  - time to complete, 5
  - what you will learn, 5
- Step 2
  - summary, 26
  - time to complete, 9
  - what it will look like, 10
  - what you will learn, 9
- Step 3
  - summary, 33
  - time to complete, 28
  - what it will look like, 28
  - what you will learn, 28
- Step 4
  - summary, 42
  - time to complete, 35
  - what it will look like, 35
  - what you will learn, 35
- Summary
  - step 2, 26
  - step 3, 33
  - step 4, 42
- Tabs
  - ToolBox, 13
  - Windows Forms, 14
- Textbox control
  - adding, 15
- Time to complete
  - step 1, 5
  - step 2, 9
  - step 3, 28
  - step 4, 35
- Toolbox
  - appendix A, 45
- ToolBox
  - about, 13
  - tabs, 13
  - viewing, 14
  - Windows forms
    - tab, 14
- Tutorial
  - at the end of, 2
  - at the end of each chapter, 2
  - getting started, 2
  - manual conventions, 2
  - source files, 2
  - using, 2
- Using the tutorial, 2
- Viewing
  - properties, 15
  - toolbox, 14
- What it will look like
  - step 2, 10
  - step 3, 28
  - step 4, 35
- What you will learn
  - step 1, 5
  - step 2, 9
  - step 3, 28
  - step 4, 35
- Windows
  - properties, 15
- Windows application
  - creating, 10
  - running, 24
- WindowsForm2
  - adding code to, 36