

Chapter 4.

SECTION I: REVOLUTION AND DESIGN ENVIRONMENT OF VISUAL BASIC 2005

Getting Started With The IDE

The easiest way to write code in Visual Basic 2005 is by using the Visual Studio 2005 **I**ntegrated **D**evelopment **E**nvironment (**IDE**). The IDE provides a wealth of features that are unavailable in ordinary text editors such as code checking, visual representations of the finished application and an explorer that displays all of the files that make up a project. An IDE is a way of bringing together a suite of tools that make developing software a lot easier.

Start Page Of IDE

The Start page appears when Visual Studio 2005 is started. The Start Page provides a centralized location to begin work using Visual Studio 2005. The Start Page can also be started by selecting **View** → **Other Windows** → **Start Page** via the menu bar. The Start page is the default home page for the Web browser in Visual Studio 2005.

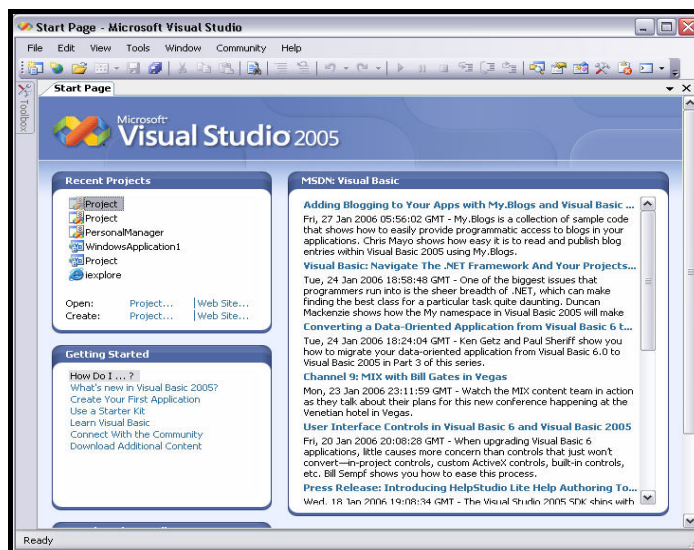


Diagram 4.1: The Start Page of Visual Studio 2005

Solutions And Projects

Visual Studio 2005 provides two containers viz. **Solutions** and **Projects** to enable the IDE to apply its wide range of tools, templates, settings and so on.

Solutions and projects contain items that represent the references, data connections, folders and files that are needed to create an application. A solution can contain multiple projects and a project typically contains multiple items.

A project includes a set of source files, its related metadata such as component references and build instructions. Projects generally produce one or more output files when built.

A solution includes one or more projects, its files and metadata that help define the solution as a whole. A solution is automatically generated when a new project is created. As and when need arises other projects can be added to the solution.

Solution Explorer provides a graphical view of the entire solution that helps managing its projects and files as the application is developed.

What Is A Solution?

Solutions manage the way Visual Studio 2005 configures, builds and deploys sets of related projects. A solution can include just one project or several projects built jointly by the development team. Sometimes a complex application might require multiple solutions

Solution allows concentrating on developing and deploying the projects, instead of sorting through all the details of managing project files, components and objects. Each solution allows:

- ❑ Working on multiple projects within the same instance of the IDE
- ❑ Working on items using settings and options that apply to an entire set of projects
- ❑ Using Solution Explorer to help develop and deploy an application
- ❑ Managing additional files opened outside the context of a solution or the project

Visual Studio 2005 stores the solution definition in two files i.e. **.sln** and **.suo**.

The **.sln** file stores the metadata that defines the solution:

- ❑ Projects associated with the solution
- ❑ Items available at solution level but not associated with a particular project
- ❑ Solution build configuration

The .suo file holds the metadata and properties to customize the IDE whenever the solution is active. For example, Solution Explorer displays a Miscellaneous Files folder for a solution if that option is enabled and the appropriate tools from the Toolbox becomes available for the type of projects included in the solution.

REMINDER



The .sln file can be shared between developers in a development team. The .suo file is a user specific file and therefore, cannot be shared between the developers of a development team.

The .suo file is a hidden file that is not displayed with the default Windows Explorer settings.

What Is A Project?

A project is used as a container within a solution to logically manage, build and debug the items that form an application.

The output of a project is usually an executable file (.exe), a dynamic link library file (.dll) or a module.

A project can be simple or complex as per the needs to meet the requirements. A simple project might consist of a form or an HTML document, source code files and a project file. More complex projects might consist of these items plus database scripts, stored procedures and references to an existing XML Web Services.

All Visual Studio 2005 development products provide a number of pre-defined project templates. Use any one of the many project templates to create the basic project container and a preliminary set of items that might be needed to develop the application, class, control or library. For example, if a Windows application option is chosen, the project offers a Windows form item. Likewise, if a Web application is chosen, the project offers a Web form item.

Each project template creates and maintains a project file to store the metadata specific to that project. This project file is created and maintained while working within the IDE. The extension of the project file and the actual content is determined by the type of project it defines.

In general, the project file stores the configuration and build settings specified for the project and its set of items. Some projects keep a list and location of the files associated with the project.

When an item is added to a project, the location of its physical source file on disk is also added to the project file. When the link is removed from the project, this information is deleted from the definition file. Each project template determines which commands are available for each item.

4. GETTING STARTED WITH THE IDE

What Are Items?

Items can be files and other parts of the project such as references, data connections or folders. In Solution Explorer, items can be organized in the following ways:

- ❑ As project items, which are items that compose a project such as forms, source files and classes within a project in Solution Explorer. The organization and display depends on the project template selected as well as any customization done
- ❑ As solution items, which are files that are applicable to the solution as a whole in the Solution Items folder of the Solution Explorer
- ❑ As miscellaneous files, which are files that are not associated with either a project or a solution but are displayed in Miscellaneous Files folder

Creating Solutions And Projects

Creating Solutions

To create a new solution, invoke Visual Studio 2005, select **File** → **New Project**. **New Project** dialog box appears as shown in diagram 4.2.1. On the left hand side of the dialog box the **Project types** are available. These project types are available for every language included in Visual Studio 2005.

On the right side the **Templates** for the selected project type are displayed. A project template helps getting started by creating a few initial files, codes and other settings for the selected project type.

Currently under **Visual Basic**, **Windows** project type is selected and the **Templates** available under the Windows project type are displayed.

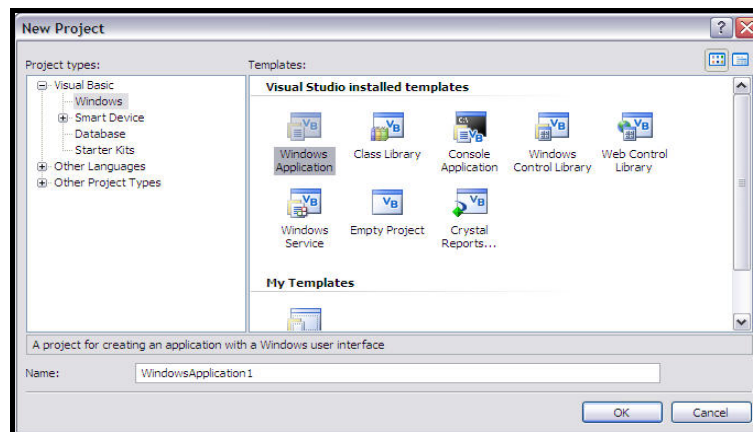


Diagram 4.2.1: New Project dialog box

Under the **Project types**, select **Other Project Types** → **Visual Studio Solutions**. Under the **Templates** select **Blank Solution** as shown in diagram 4.2.2

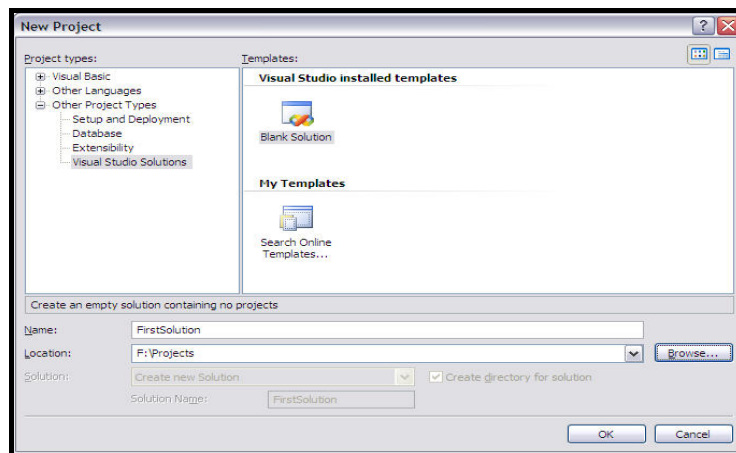


Diagram 4.2.2: Selecting Blank Solution

Enter an appropriate name for the solution. If necessary change the location of the solution directory by selecting **Browse...** and then specify a new location. Click **OK** when done.

After creating an empty solution, new or existing projects and items can be added to an empty solution by using **Add New Item** or **Add Existing Item** from the standard toolbar.

Creating A Project

To start with a new project, select **File** → **New Project**. **New Project** dialog box appears as shown in diagram 4.2.3.

Currently Under **Visual Basic**, **Windows** project type is selected and the **Templates** available under the Windows project type are displayed on the right side.

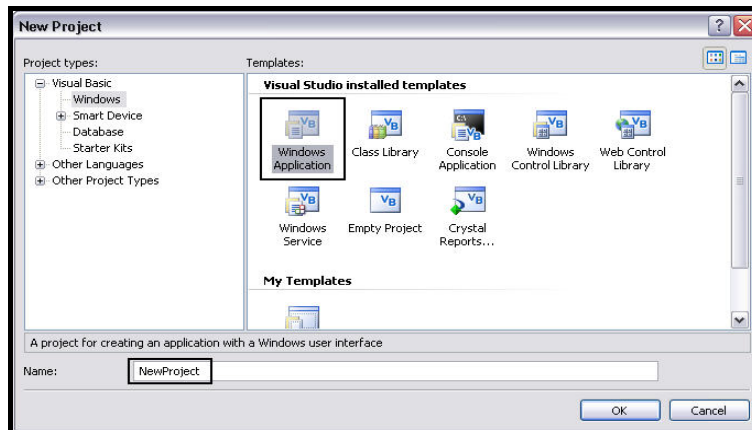


Diagram 4.2.3: New Project dialog box

4. GETTING STARTED WITH THE IDE

Whenever a new project is created and there is no pre-created solution, Visual Studio 2005 creates one and the project is held inside this solution.

Select **Windows Application** available under **Templates** as shown in the diagram 4.2.3.

REMINDER



A Windows Application has a **Graphical User Interface**, which is often referred to as **GUI**. A Windows Application is an application, which is displayed in a window. Some of the examples of Windows Applications are Notepad, Microsoft Word and so on.

By default, all Windows projects created are named **WindowsApplication** along with a running number as shown in diagram 4.2.1. This can be changed if desired. Once done, click OK.

The IDE after a project is created, is as shown in the diagram 4.3.

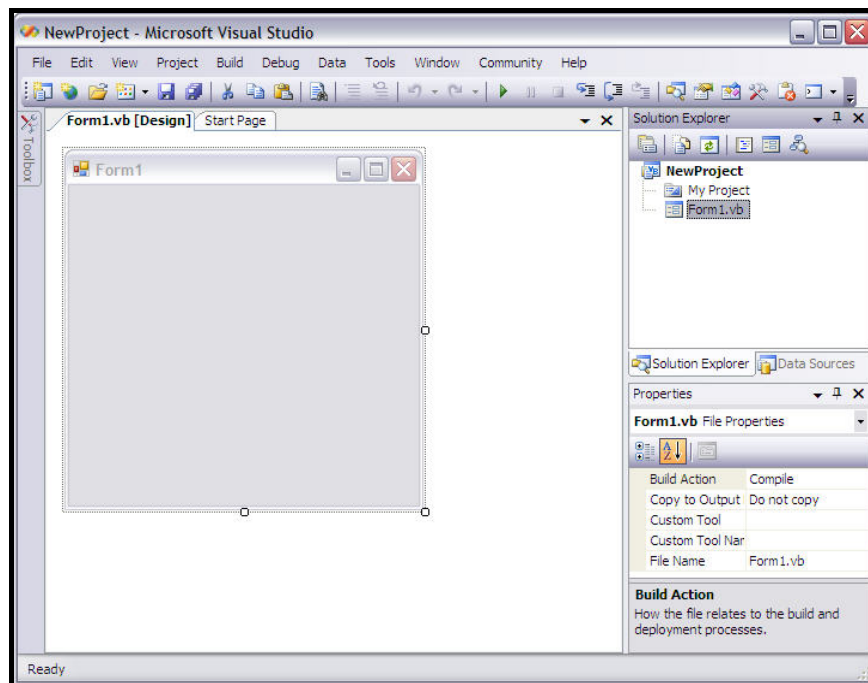


Diagram 4.3: The IDE after project creation

Working With Solution Explorer

The **Solution Explorer** is used to manage the projects that make up a solution and the files that make up each project as shown in diagram 4.4.1

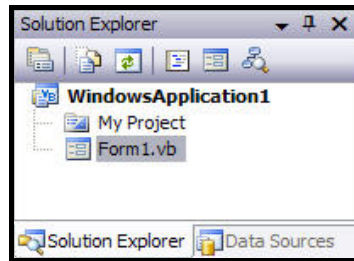


Diagram 4.4.1: The Solution Explorer window

When a file is double-clicked in the Solution Explorer, the file opens in Form Designer view. Using Solution Explorer, working on multiple files is also possible. The Solution Explorer also enables working with files known as miscellaneous files, which are not a part of any project.

Using Solution Explorer files can be opened, managed, added and removed. A context menu appears when a file inside the solution explorer is right clicked. The context menu contains options, which can perform various operations such as copying, cutting, deleting and so on. Additionally, it also allows removing an item from a project by selecting the **Exclude From Project** option from the context menu.

The options in the context menu vary depending on the object selected. For example, the options that appear in the context menu for a form are different from the options that appear for other items.

To change the name of an object using **Solution Explorer**, right click an object and select **Rename** option from the context menu as shown in diagram 4.4.2. Rename the object and press **Enter** as shown in diagrams 4.4.3 and 4.4.4.

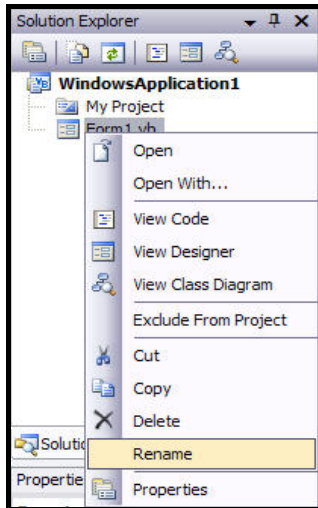


Diagram 4.4.2: Context Menu

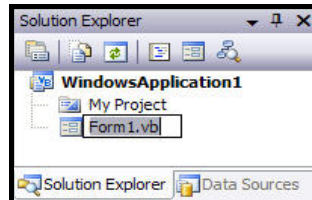


Diagram 4.4.3: Renaming the form

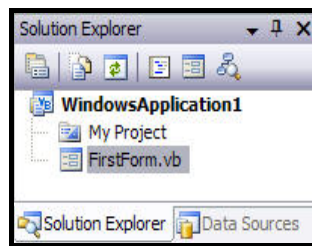


Diagram 4.4.4: The form renamed

4. GETTING STARTED WITH THE IDE

Working With The Properties Window

The Properties window is used when the appearance or the operation performed by a form or a control needs to be changed. If any particular control such as button on a form or the form itself is selected, the Properties window is populated with the respective object's properties. For example, double click a form to have its properties populated in the Properties window as shown in diagram 4.5.1.

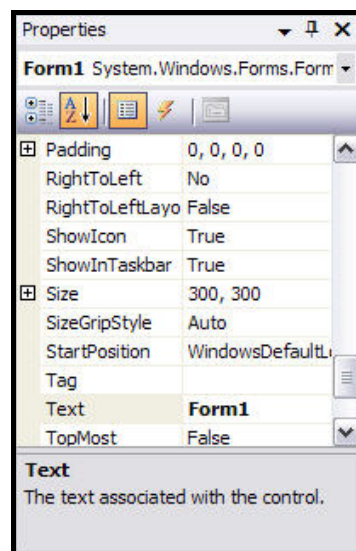


Diagram 4.5.1: The Properties window

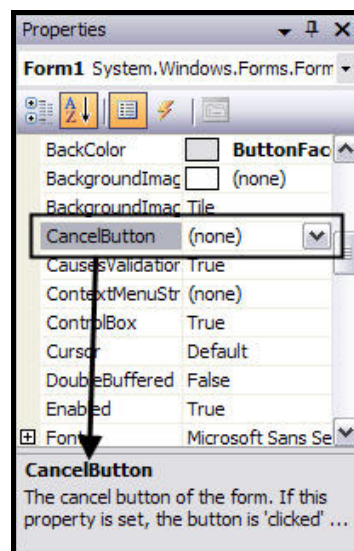




Diagram 4.5.2: The description of every property

The Properties window, allows editing the properties of the object selected.

The properties displayed in the Properties window are different for different objects i.e. the properties of a button will differ from the properties of a form.

The Properties window displays a toolbar right after the objects name. This toolbar holds the following buttons:

-  (Categorized) - It groups the properties of a control based on categories. For example, when a button is selected and the  is clicked, the properties of the button control are grouped into categories such as Appearance, Data, Layout and so on

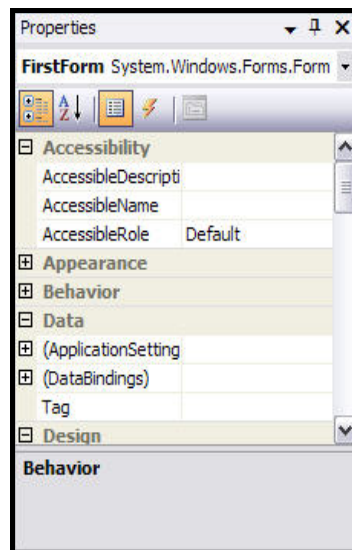





Diagram 4.5.2.1: Properties grouped on categories

- ❑  (Alphabetical) - It arranges the properties alphabetically
- ❑  (Properties) - It shows a list of properties for a particular form or controls as shown in diagram 4.5.2
- ❑  (Events) - It shows a list of events for a particular form or controls as shown in diagram 4.5.3

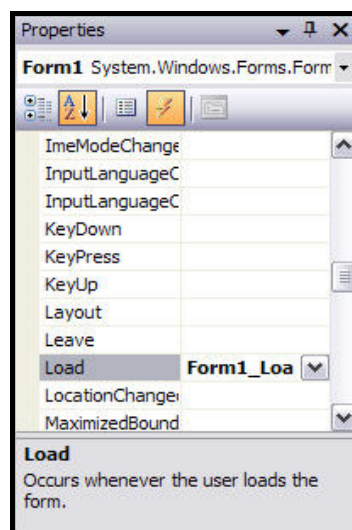



Diagram 4.5.3: The events

4. GETTING STARTED WITH THE IDE

-  (Property Pages) - It displays the Property pages dialog box for the selected component. It can be used for viewing and editing properties related to the configuration of the project

The Properties window for each form or control has two sections i.e. the name of the property and the value of the property.

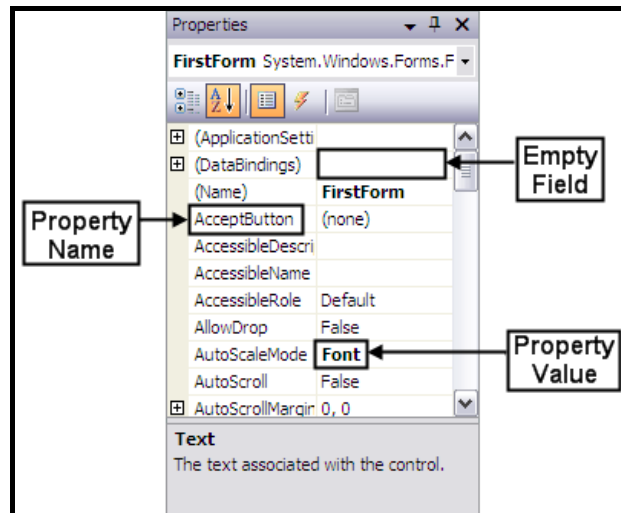


Diagram 4.5.3: Sections in Properties windows

The name of a property appears in the left column. The names of properties are always one word. This same name can be used to access the property via code.

The box on the right hand side of each property name represents the value held inside that property. This can be set using this window.

The different fields available in the Properties windows are:

- **Empty Fields** - By default, the empty fields have nothing in their properties as shown in diagram 4.5.3. Most of these properties are dependent on other settings of the program. For example, a menu property for a form can be set only after creating a menu. To set the property value for such a field, just type the property value or select it from a list if one exists
- **Text Fields** - There are fields that expect the developer to type a value as shown in diagram 4.5.4. Most of these fields have a default value. To change the value of the property, click the name of the property, type the desired value and press **Enter**. While some properties, such as the Text, would allow anything, some other fields expect a specific type of text such as a numeric value

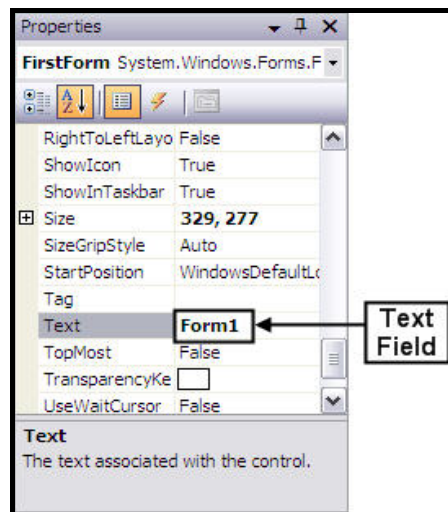


Diagram 4.5.4: Text field

- **Expandable Fields** - Some fields have a + (plus) sign, which indicate that property has a set of sub properties that actually belong to the same property and are set together. To expand such a field, click the + (plus) sign and a - (minus) sign will appear as shown in diagram 4.5.5.1 and 4.5.5.2

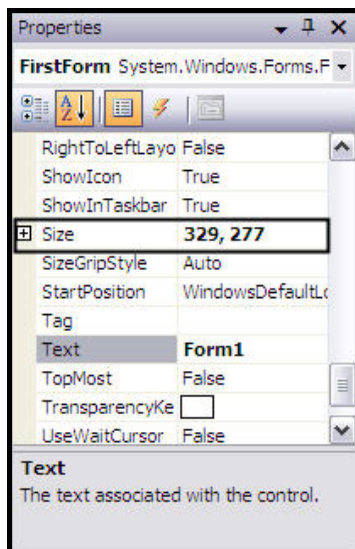


Diagram 4.5.5.1: Clicking + sign

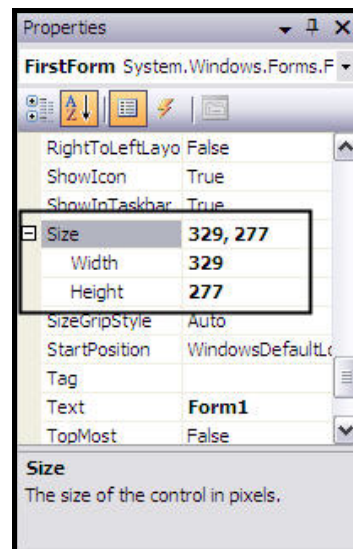


Diagram 4.5.5.2: Field Expanded

Some of the properties are numeric such as Size as shown in diagram 4.5.5.1. With such a property, click its name and type two numeric values separated by a comma.

4. GETTING STARTED WITH THE IDE

Some other properties hold an enumerator or a class. If such a field is expanded, it will display various options. For example, with Font property, selection can be made from a list.

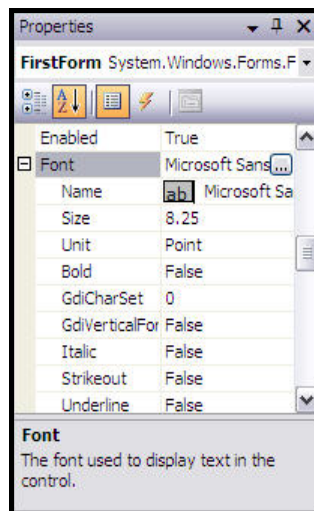


Diagram 4.5.6: Font property expanded

- **Boolean Fields** - Some fields can have only Boolean values i.e. either True or False value. To change their setting, either select from the combo box or double-click the property to toggle to the other value

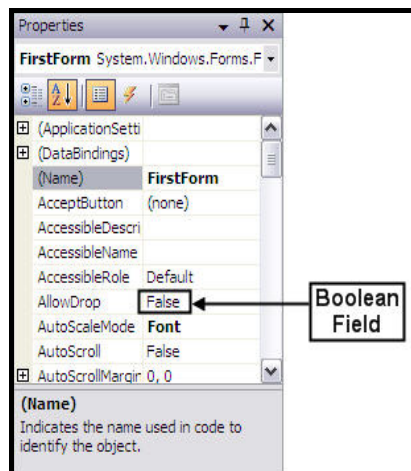


Diagram 4.5.7: Boolean field

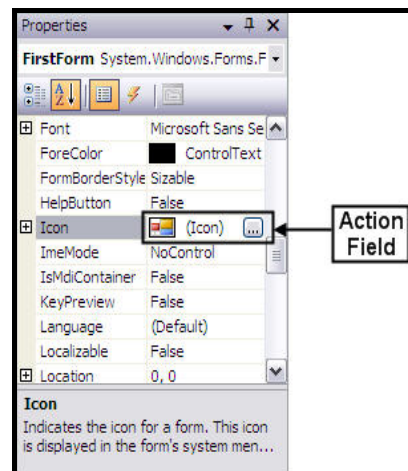


Diagram 4.5.8: Action Field

- **Action Fields** - Some fields would require a value or item that needs to be set through an intermediary action. Such fields display an ellipsis button. When this button is clicked, a dialog box pops up and the value can be set using this dialog box

- ❑ **Selection Fields** - The value of some fields can be changed using a drop down list box as well. This drop down list box will display a list of property values that can be set as shown in diagram 4.5.9

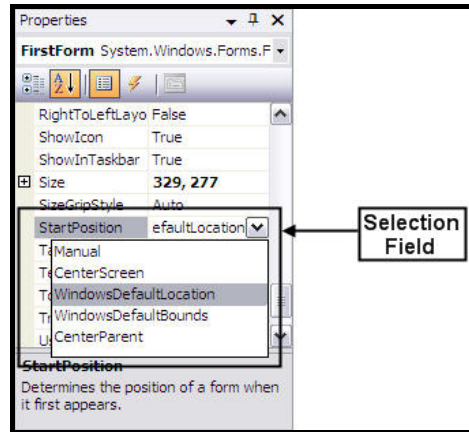



Diagram 4.5.9: Selection Field

There are various types of selection fields. Some of them display just two items. To change their value, just double-click the field to toggle between the two values. Some other fields have more than two values in the field. To change them, just click the adjoining arrow and select the desired value from the list or double-click the property name a few times until the desired value is selected.

It is a good programming practice to change the name of the form in the title bar i.e. . Select the **Text** **Form1** property from the Properties window and rename Form1 to the appropriate form name as shown in diagram 4.5.10. Press **Enter** and the name changes according to the new name entered.

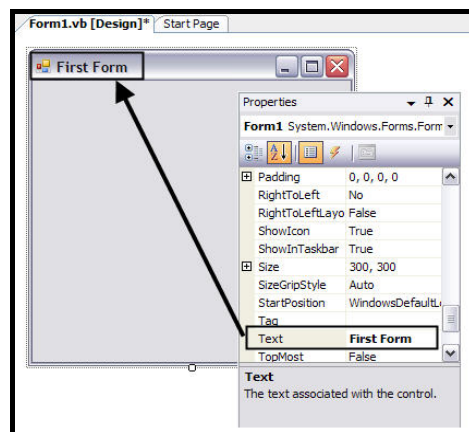


Diagram 4.5.10: The name in the title bar changed

4. GETTING STARTED WITH THE IDE

Working With Toolbox

The Toolbox contains various tools and controls available in Visual Basic 2005. To open the Toolbox, move the mouse pointer on the Toolbox tab displayed on the left margin of the IDE. Alternatively, open the Toolbox by selecting the Toolbox command from the View menu.

By default, the Toolbox displays only the **General** tab. However, the Toolbox displays additional tools depending on the currently opened designer or editor. To view all the tools in a particular tab in the Toolbar, click the **+** (plus) sign available on the left of each tab in the Toolbox.

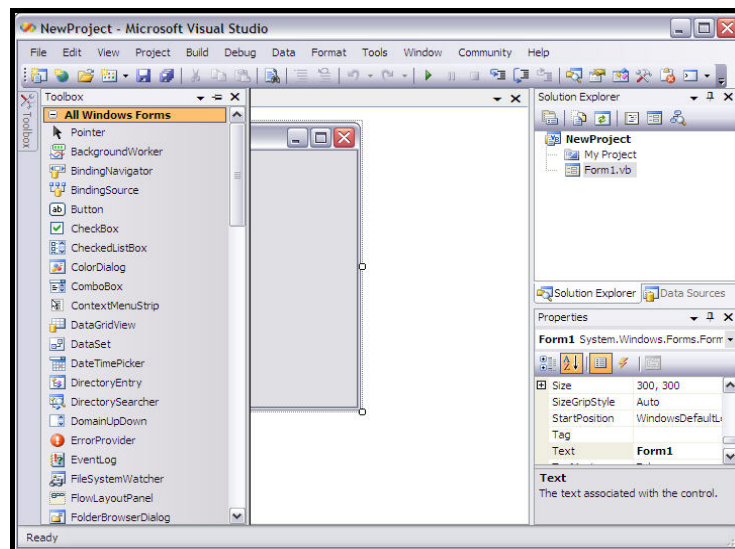


Diagram 4.6.1.1: The Toolbar loaded with controls and items

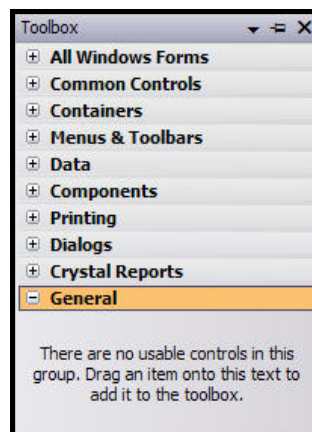


Diagram 4.6.1.2: The Toolbar tabs

The following are some of the tabs (right at the bottom of the toolbox) available in the Toolbox as shown in diagram 4.6.1.2:

- ❑ **General tab** - By default, the General tab does not display any controls. However, controls can be added such as custom controls to the General tab. Custom controls are controls created by users or third-party vendors

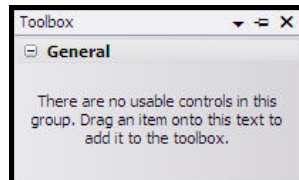


Diagram 4.6.2: The General tab

- ❑ **Crystal Reports tab** - It displays components such as CrystalReportViewer, ReportDocument and so on, that can be used in Crystal Reports

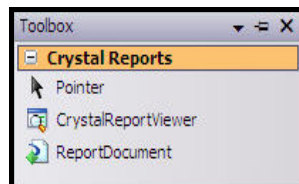


Diagram 4.6.3: The Crystal Reports tab

- ❑ **Dialogs tab** - It displays components such as ColorDialog, FontDialog and so on, that can be used for Dialog boxes

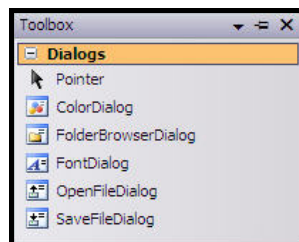


Diagram 4.6.4: The Dialogs tab

- ❑ **Printing tab** - It displays components such as PrintDialog, PrintDocument and so on, that are related to printing

4. GETTING STARTED WITH THE IDE

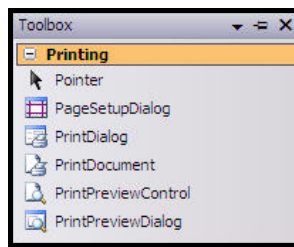


Diagram 4.6.5: The Printing tab

- **Components tab** - It displays components such as EventLog, MessageQueue and so on, that can be added to Visual Basic 2005 projects

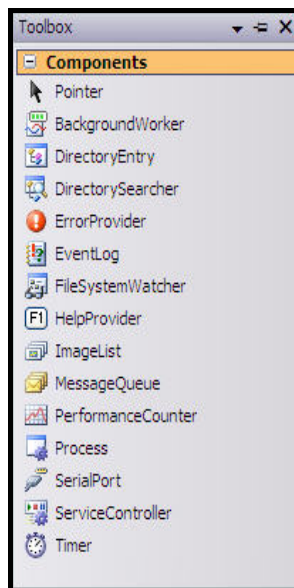


Diagram 4.6.6: The Components tab

- **Data tab** - It provides data objects such as DataSet, DataGridView and so on, that can be included in Visual Basic 2005 forms and components

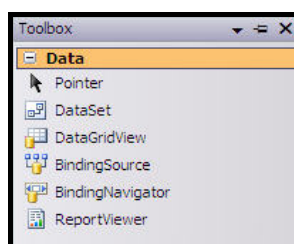


Diagram 4.6.7: The Data tab

- ❑ **Menus & Toolbars tab** - It provides components such as ContextMenuStrip, ToolStrip and so on, which are the types of menus or toolbars to be included to Visual Basic 2005 projects

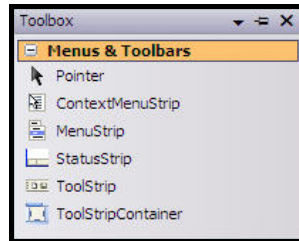


Diagram 4.6.8: The Menus & Toolbars tab

- ❑ **Containers tab** - It provides components such as Panel, TabControl and so on, which are different types of layout for the look and feel of Visual Basic 2005 forms

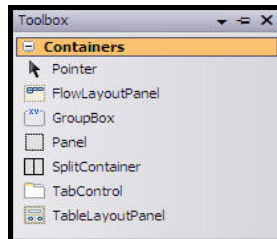


Diagram 4.6.9: The Containers tab

- ❑ **Common Controls tab** - It displays common controls that can be added to forms

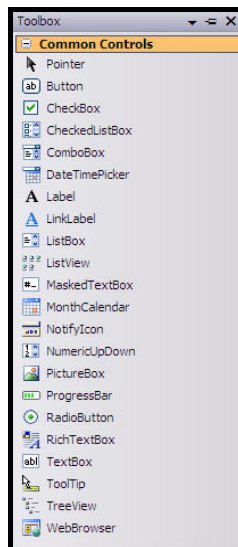


Diagram 4.6.10: The Common Controls tab

4. GETTING STARTED WITH THE IDE

- **All Windows Forms tab** - It provides all the components available in the tabs Common Controls, Dialogs, Printing, Data, Components, Containers, Crystal Reports and Menus & Toolbars

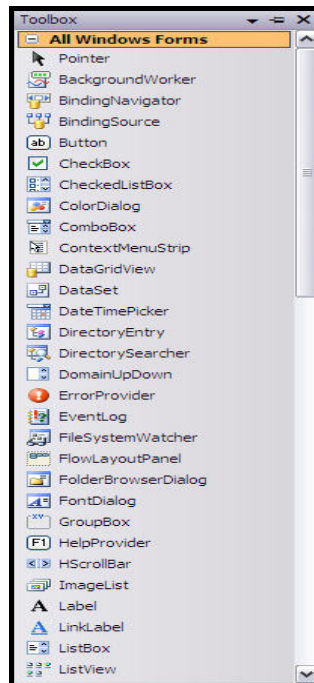


Diagram 4.6.11: The All Windows Forms tab

There are three ways to add controls to the form:

- Click once on any control that is to be added to the form. The mouse pointer changes to represent a button and now click once on the form. The button is added to the form. Refer to diagrams 4.7.1 and 4.7.2

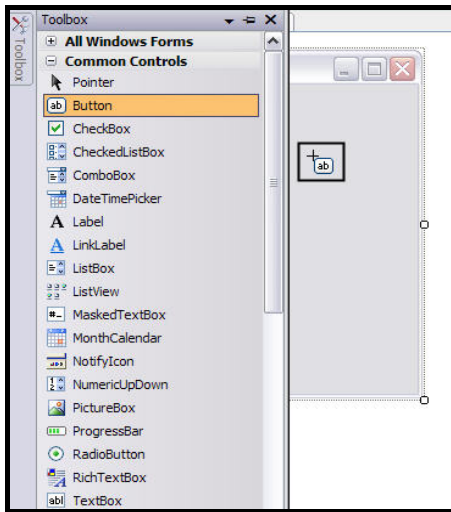


Diagram 4.7.1: Single clicking Control

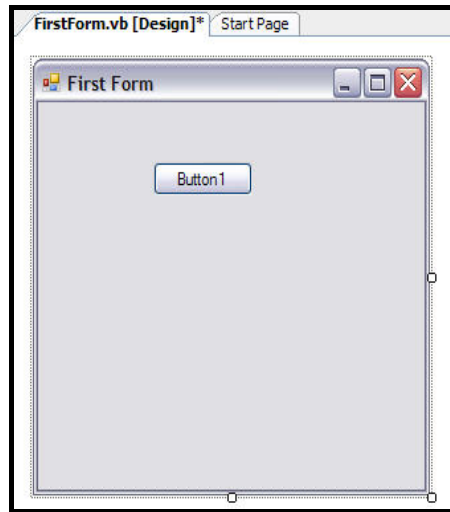


Diagram 4.7.2: The button added

- The other way is the drag and drop method to add the controls to the form. Refer to diagrams 4.7.3 and 4.7.4. Here the mouse pointer represents the combobox

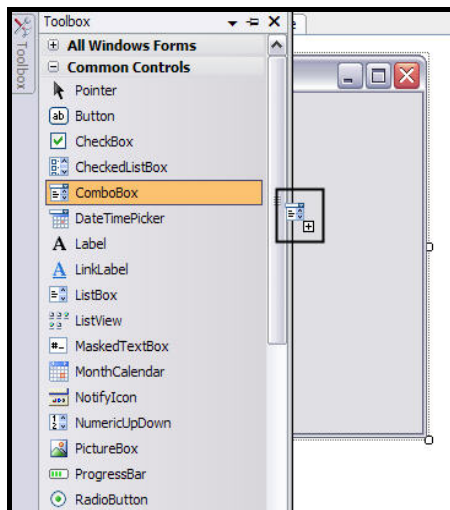


Diagram 4.7.3: Dragging and dropping Control

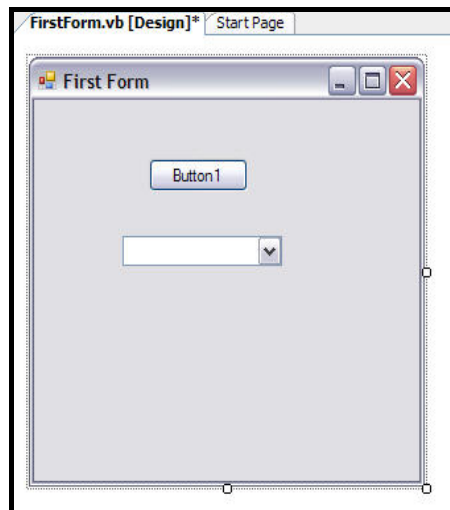


Diagram 4.7.4: The combobox added

4. GETTING STARTED WITH THE IDE

- Double clicking a control is the third way of adding a control to a form. Here the mouse pointer does not change instead the control is immediately added to the form. The control gets added at the top left corner just below the title bar as shown in diagrams 4.7.5 and 4.7.6

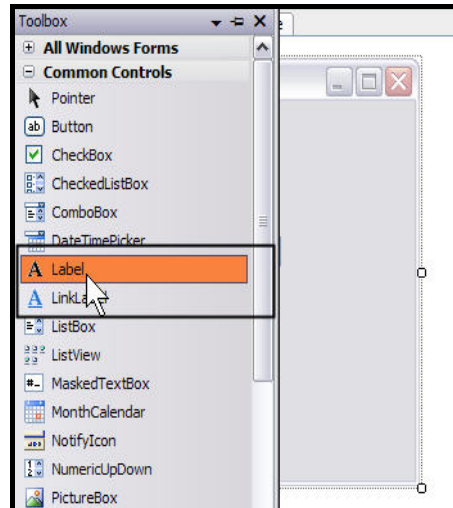


Diagram 4.7.5: Double clicking Control

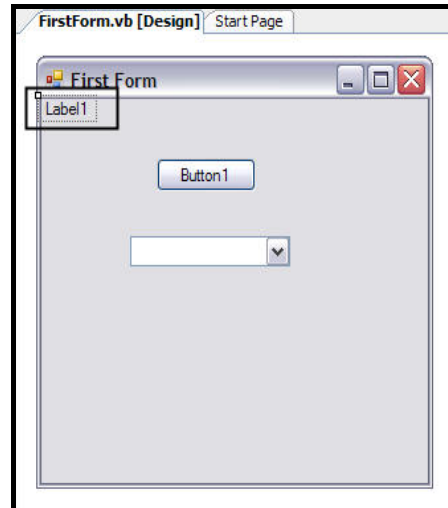


Diagram 4.7.6: The Label added

Naming Conventions

By default Visual Basic 2005 IDE names the controls. For example, Label1 for the first Label control, Button2 for the second Button control, and so on. Each name should be changed and prefixed with a short identifier describing the type of control it is. This makes it much easier to understand what type of control is being dealt with when the code is referenced later.

For example, there is a control called **Name**, without a prefix of lbl or txt, the developer would not know whether it is a textbox or label with that name.

When multiple developers work together on a particular project, it is very important to keep the coding style consistent. One of the most commonly used styles used for naming controls within an application is the Modified Hungarian notation.

The notion of prefixing control names to identify their use was brought forth by Dr. Charles Simonyi. He worked for the **Xerox Palo Alto Research Center** (XPARC), before joining Microsoft. He came up with short prefix mnemonics that allowed developers to easily identify the type of information a variable might contain.

Since Dr. Simonyi is Hungarian and the prefixes make the names look a little foreign, the name Hungarian Notation stuck. Since the original notation was used in C/C++ development, the notation for Visual Basic 2005 is termed Modified.

Some of the prefixes, which can be used in Visual Basic 2005 are as follows:

Control	Prefix	Example
Button	btn	btnName
ComboBox	cmb / cbo	cmbProfession
CheckBox	chk	chkSelect
Label	lbl	lblState
RadioButton	rb / rdb	rbSingle
TextBox	txt	txtCountry
PictureBox	pic	picPersonel
Grid	grd	grdData

Hungarian Notation can be a real time-saver when looking at code written by someone else or at the code written several months ago. However, the most important thing is, to be consistent in the naming. When starting with coding, pick a convention for naming. It is recommended that the de facto standard, Modified-Hungarian, is used for Visual Basic 2005, but it is not mandatory.

Once a convention is picked, stick to it. When modifying someone else's code, use theirs. There is very little code that is ever written, put into production and then forgotten. A standard naming convention followed throughout a project will save countless hours when the application requires maintenance later.

Designing The Form

If the double click method is used to add a control, then the control is situated at top left control of the form or if the drag and drop method is used, then the control is placed accordingly at the place appointed.

In either way, the control can be repositioned. To position the control, place the mouse pointer over the control to be repositioned. The mouse pointer changes its shape to that of four arrows as shown in diagram 4.8.1.

Then click on the left mouse button and drag the control to another location and release the left mouse button after the desired position is selected.

4. GETTING STARTED WITH THE IDE

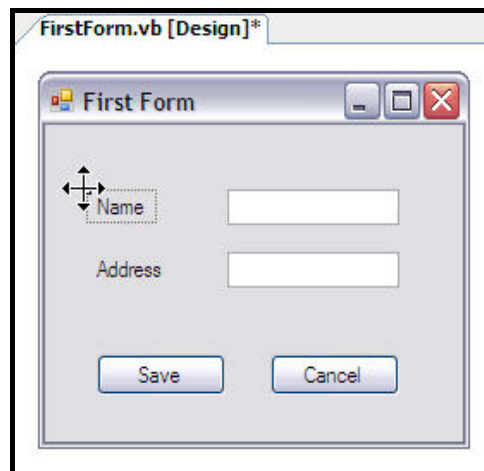


Diagram 4.8.1: Changing the position of a control

When a TextBox control is dropped or selected/clicked on the form a black arrow can be seen on the top right side of the text. When this arrow is clicked, Visual Basic 2005 shows the option of multiple lines. If multiple lines are needed, then tick the checkbox available next to it.

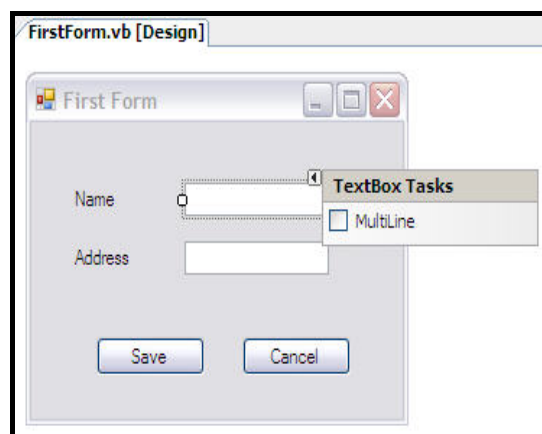


Diagram 4.8.2: The option of multiple lines

These options differ for different controls. For example, when a **RichTextBox** control is dropped then its options are **Edit Text Lines** and **Dock In Parent Container**.

Diagram 4.8.3 shows the labels, textboxes and buttons in a very crude manner. That is they are not in one line or not adjusted according to their counterparts. The label Name is somewhere while its counterpart textbox is somewhere else.

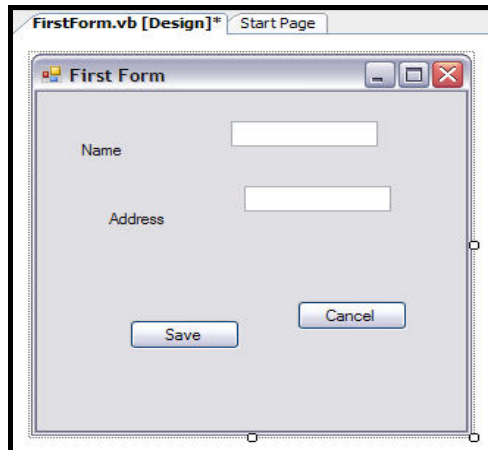


Diagram 4.8.3: Controls in a crude manner

Visual Basic 2005 IDE has a new feature to adjust controls with respect to each other in forms.

For example, to bring the Name label and its textbox in one line click on any of the control and accordingly adjust the control by dragging it in a direction as desired as shown in diagrams 4.8.4 and 4.8.5.

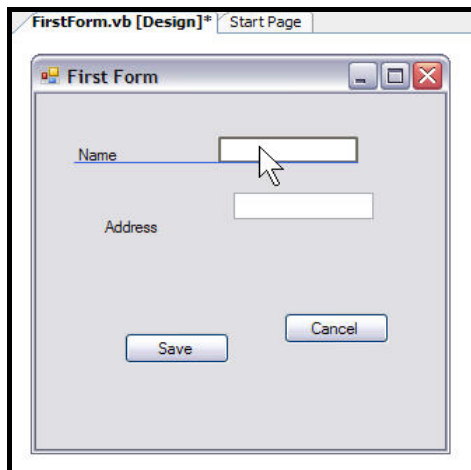


Diagram 4.8.4: Double clicking Control

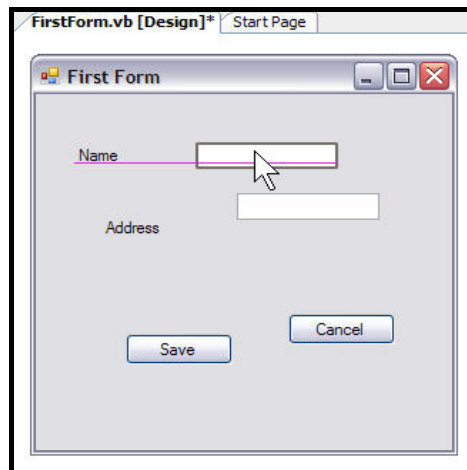


Diagram 4.8.5: The Label added

While dragging the control, the **blue line** indicates that the base of the textbox control is exactly inline with the base of the label control where as the **purple** line indicates that text box control is exactly horizontally centered inline with the label control.

4. GETTING STARTED WITH THE IDE

The diagram 4.8.6 shows two blue lines and a purple line. Two blue lines indicate that the text box of Address is vertically inline with the textbox of Name from both sides. A purple line indicates that text box control is exactly horizontally centered inline with the Address label control.

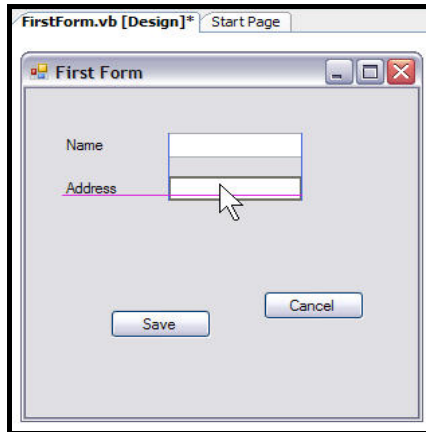


Diagram 4.8.6: Multiple lines while adjusting controls

Resizing a form or a control is also possible. For that select a form or any control whose size has to be resized. Keep the mouse pointer to any small round spot, which can be seen on the borders. The mouse pointer changes, now drag and change the size of the form or controls as desired as shown in diagrams 4.8.7 and 4.8.8.

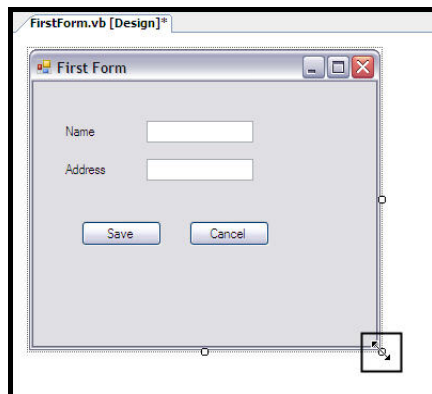


Diagram 4.8.7: Mouse pointer changed

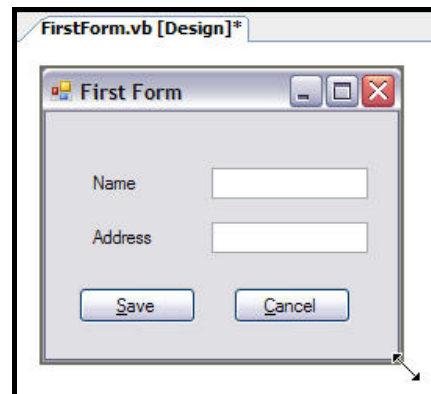


Diagram 4.8.8: Dragging the mouse pointer

If a shortcut key or a hot key has to be created for a button, add an ampersand (&) in the Text property of that button. The letter with the & sign placed in front of it will become underlined as shown in diagram 4.8.9. This indicates to users that they can use **Alt + Letter** key combination.

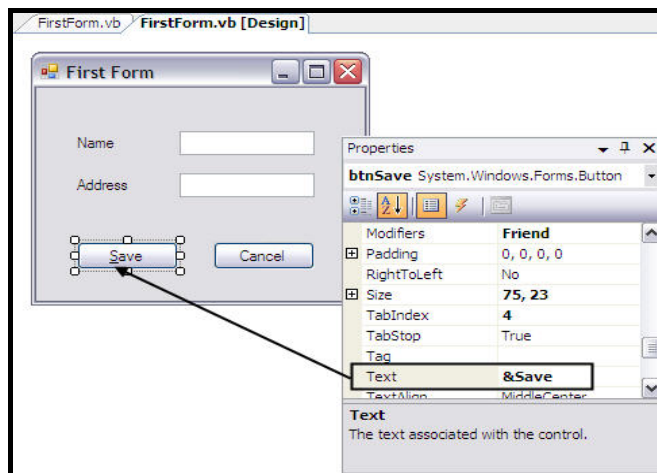



Diagram 4.8.9: Creating Shortcut key

Pressing **Alt + S** would be the same as clicking the **Save** button as shown in diagram 4.8.9. There is no need to write code to make a shortcut key.

WARNING

 Sometimes when the form is run, the underline may not appear unless and until the Alt key is pressed.

Short Description Of Toolbar

Many toolbars are available within the Visual Studio 2005 IDE such as Formatting, Help, Style Sheet, Layout and so on. Each of these toolbars can be added to or removed from the IDE using **View → Toolbars** menu item.

When a new project is selected, by default the standard toolbar appears. The **standard** toolbar includes standard window toolbar buttons such as Open, Cut, Copy, Paste and so on. On the right hand side of the standard toolbar, there are several buttons that call other windows in the IDE. Additional toolbars dynamically appear depending on the function performed.

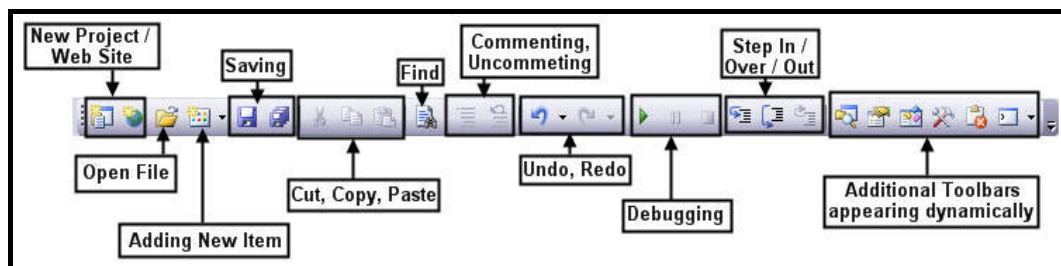






Diagram 4.9: Standard Toolbar


4. GETTING STARTED WITH THE IDE


 provide access to creating new project or web site, open the commonly used projects, adding new items and saving new project or the existing project.


 are used for editing i.e. cut, copy and paste.

 allows finding text in the code throughout the code or project.

 allow commenting and uncommenting code.  is for undoing and redoing edits.

 provides the ability to run the form, pause the execution at runtime and terminate its execution.

 are icons to step in or step out or step over code at runtime for debugging.

Lastly  provides quick links to the Solution Explorer window, Properties window, Object Browser, Toolbox, Error List, Command window and many other windows. The other window's icons appear as the work on the project progresses.

Saving The Project

To save the project, select **File** → **Save All**. **Save Project** dialog box appears as shown in diagram 4.10.1.

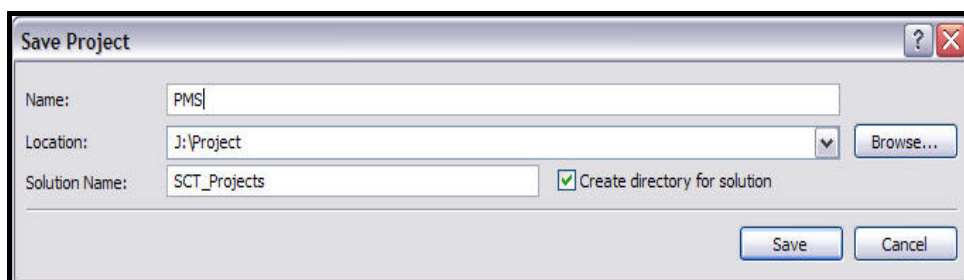
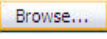


Diagram 4.10.1: Save Project dialog box

This dialog box displays the name of the project entered earlier while creating a new project. The Solution Name by default is assigned the same name as that of the project only if it was not saved earlier using a different name.

By default the Location i.e. the path where the project is located will be **C:\Documents and Settings\<Name>\My Documents\Visual Studio 2005\Projects**. To change the current path click  available next to **Location** text box and select the desired path for saving the project.

After saving, the directory structure that holds the project saved will look as shown in diagram 4.10.2.1 and 4.10.2.2.

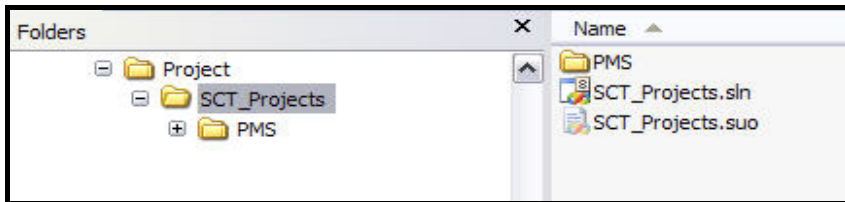


Diagram 4.10.2.1: Directory Structure of the project saved

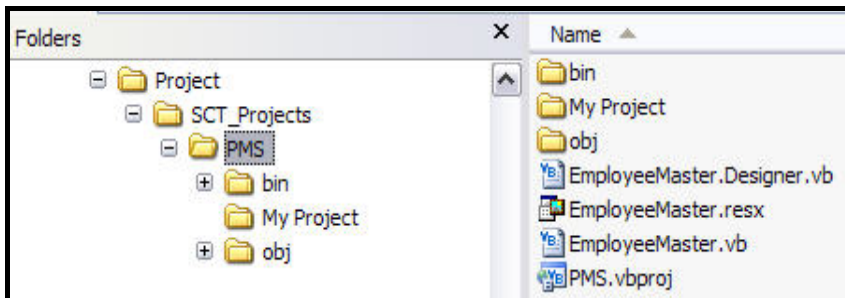


Diagram 4.10.2.2: Directory Structure of the project saved

The following is the directory structure:

- **J:\Project** is parent directory that holds the solution named **SCT_Projects**
 - **SCT_Projects** is the main solution directory where all the projects are in form of directories. This directory holds two files and one directory named **PMS**:

The Files:


- **SCT_Projects** with the extension Microsoft Visual Studio Solution (.sln) is the solution file that stores the metadata to define the solution
- **SCT_Projects** with the extension Microsoft Visual Studio User Options (.suo) holds the metadata and properties to customize the IDE

The Project Directory:

- **PMS** is the project directory where all the forms are saved along with project files, designer files and so on
 - **EmployeeMaster.vb** is the file, which holds the code written for that particular form. If the file **EmployeeMaster.vb** is opened then only the code held inside that file will be displayed
 - **EmployeeMaster.Designer.vb** is the file generated by the IDE. There are designer files for every form created. This file holds the actual form design layout
 - **EmployeeMaster.resx** is the .NET resource file which stores information about various resources used in the project

4. GETTING STARTED WITH THE IDE

- **PMS.vbproj** is the project's main file that stores the metadata to define the project
- The **bin** directory holds all the runtime executables
- The **My Project** directory holds project related files that define and hold assembly information, settings related to that project and resource files
- The **obj** directory holds any assemblies created as a result of adding a COM reference to the project

If any changes are made either in the code editor or in form design, Visual Basic 2005 displays an indication of the changes made and not saved in the tabbed window. It displays * (asterisk) as an indication. Once the project is saved using **Ctrl + S** or clicking , this indication will be removed.

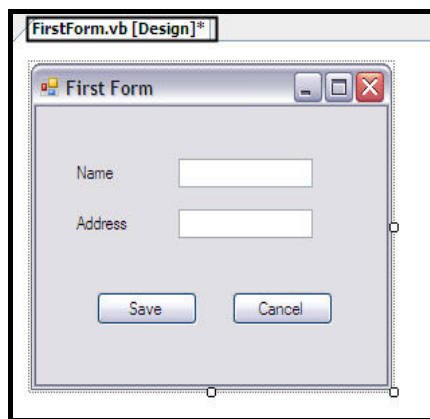


Diagram 4.10.3: Asterisk (*) indication for saved project

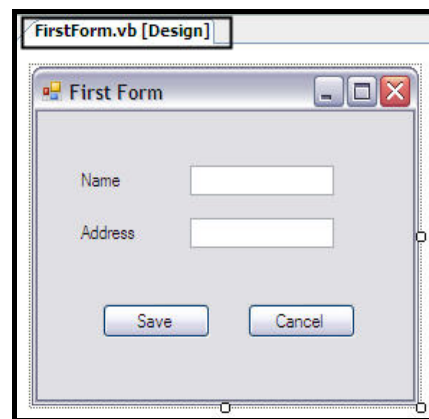


Diagram 4.10.4: Asterisk (*) removed after the project is saved

Opening The Existing Project

Projects that already exist can be opened using one of the following ways:

Using Start Page

The Start Page provides a section as shown in diagram 4.10.5.1 that lists the most recent projects, select the desired project from the list to open.



Diagram 4.10.5.1: Recent Projects

The Start Page also provides **Open: Project... | Web Site...** via which a project that is not listed in the **Recent Project** list can be opened. To do so, click **Project...**. This will open **Open Project** dialog box as shown in diagram 4.10.5.2.

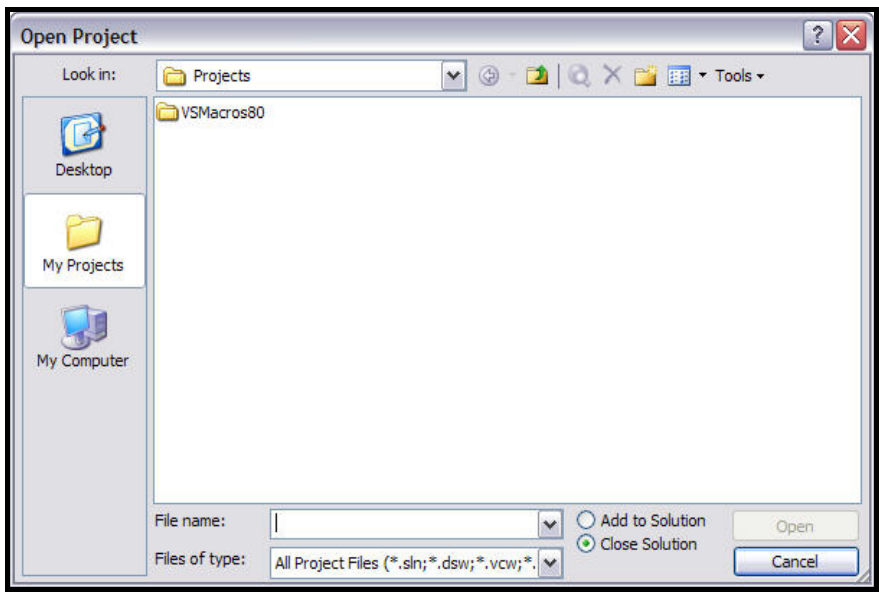


Diagram 4.10.5.2: Open Project dialog box

Using this dialog box locate the desired project and click **Open**.

4. GETTING STARTED WITH THE IDE

Using Menu Item

To open a project using a menu item, select **File** → **Open Project** or use **Ctrl + O**.

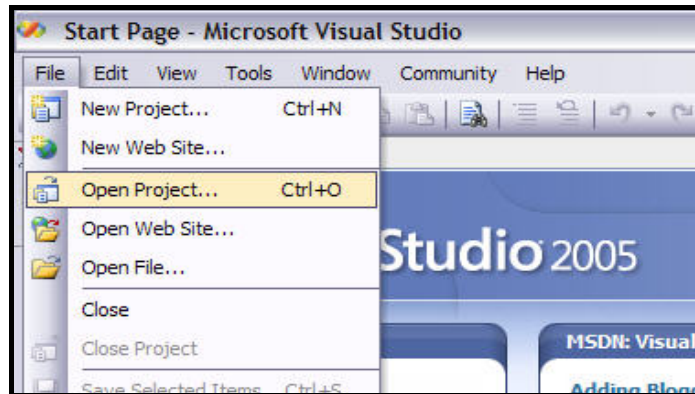


Diagram 4.10.5.3: Opening a project using Menu items

This will display the **Open Project** dialog box as shown in diagram 4.10.5.2. Using this dialog box locate the desired project and click .

Using The Toolbar

To open a project using a toolbar, click .

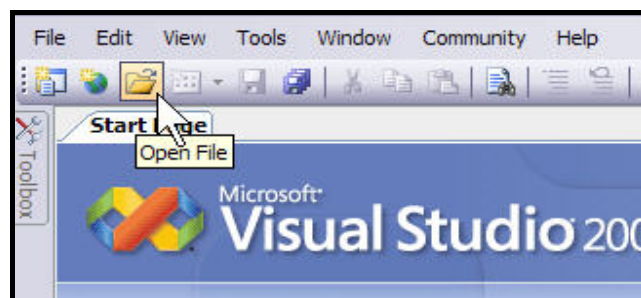


Diagram 4.10.5.3: Opening a project using Toolbar


This will display the **Open Project** dialog box as shown in diagram 4.10.5.2. Using this dialog box locate the desired project and click .

Alternatively, use Windows Explorer, locate the desired project and double click the **.sln** file to open a solution that holds the project or double click the **.vbproj** file to directly open the project.

Using The Code Editor

The Code Editor window allows creating and editing Visual Basic 2005 source code. After designing the user interface for the project by placing controls on the form, switch to the Code Editor to develop the Visual Basic code spec that makes the forms and their controls functional.

The easiest way to call up the Code Editor is to double-click a form or a control. Then, begin typing the statements that will be executed when the user performs the most common action on that control.

Another way to call up the Code Editor is to click  available in the Solution Explorer.

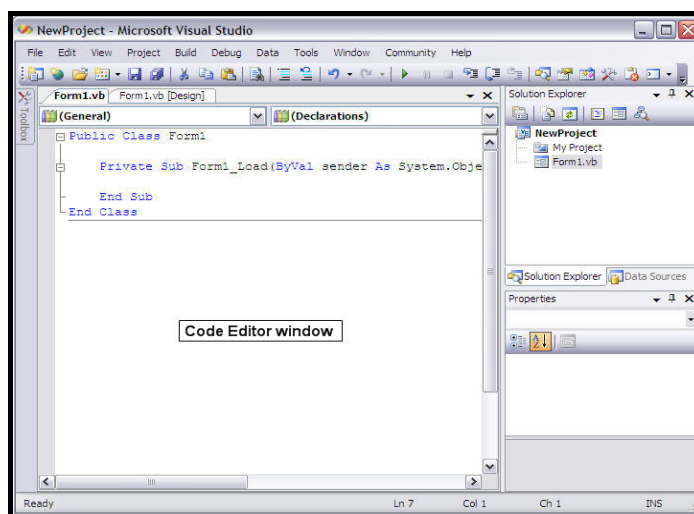


Diagram 4.11.1: Code Editor Window

If a form is double clicked then the procedure with the form name appears in code editor window held inside the class declaration. The procedure is a part of that class.

The Code Editor works much like other text editors. However, the Code Editor has a number of special features that simplify the task of editing / writing Visual Basic code:

- ❑ **Color** - Color is used to distinguish Visual Basic 2005 keywords from variables, comments and other language elements. And also many types of coding errors are automatically highlighted as they are typed so they can be corrected or rectified

4. GETTING STARTED WITH THE IDE

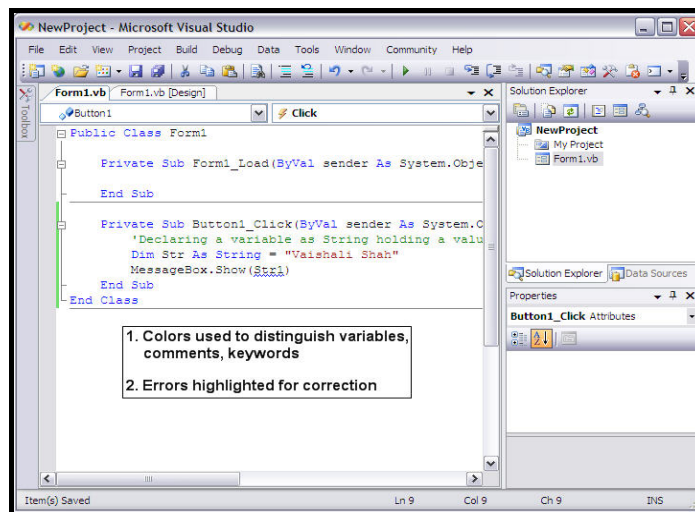


Diagram 4.11.2: Colors for distinguishing keywords and errors highlighting

- **IntelliSense** - This feature pops up a drop down list automatically and guides the developer by showing what methods / properties / data types and so on are available for a given object. As the keywords are typed letter by letter it reaches the matching word. For example, if String is the word to be typed, just type Str and IntelliSense will automatically highlight that keyword into its list and also display the details of that keyword as a tool tip. Either select this keyword by pressing Enter or type the entire keyword

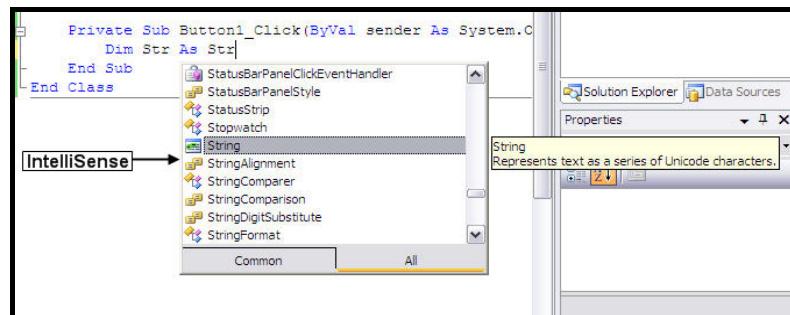


Diagram 4.11.3: IntelliSense

- **Parameter Information** - Parameter Information displays the parameters that a procedure, function, method or property accepts. If a function is used in the source code named `MessageBox.Show`, after the function name is typed into the code editor window and as soon as a left parenthesis is typed to begin describing parameters, Visual Basic 2005 automatically shows the correct parameters along with data types and overloaded versions of that functions if any (represented in this case as 1 of 21 that shows that `MessageBox.Show` function has 21 overloaded versions)

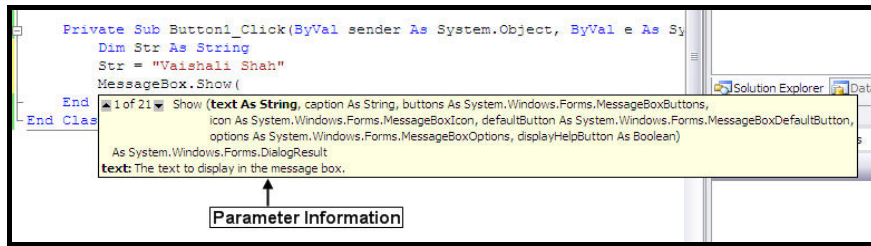


Diagram 4.11.4: Parameter Information

- **Description of an Error** - To see the description of an error, move the mouse pointer over the error (represented as a curly blue line)

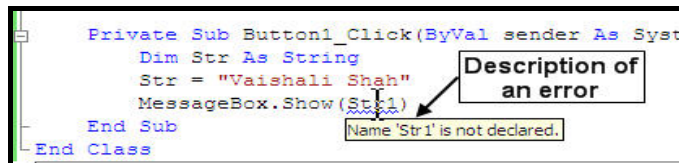


Diagram 4.11.5: Description of an error

- **Detection Of Wrong / Outdated Syntax** - If the syntax is wrongly written or belongs to some previous version which Visual Basic 2005 does not support then Visual Basic 2005 IDE underlines the error with an exclamation mark as shown in diagram 4.11.6.1. Here **Type...End Type** syntax has been replaced by the **Structure...End Structure** syntax. When the mouse pointer is moved over the underlined word it displays an exclamation mark as shown in diagram 4.11.6.1

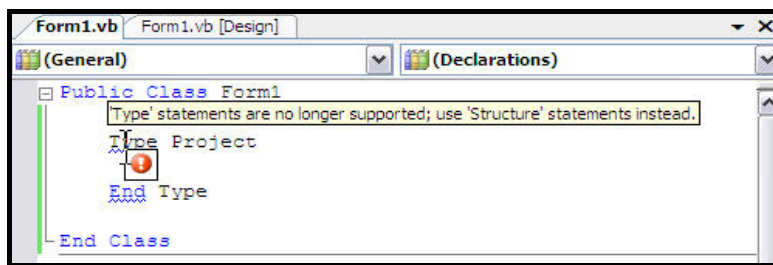


Diagram 4.11.6.1: Warning sign and a short description of error

If the mouse pointer is moved over the warning sign a down arrow appears. Click once on the arrow and a description of syntax appears that informs that the Type syntax is no longer supported use Structure syntax instead as shown in diagram 4.11.6.2

4. GETTING STARTED WITH THE IDE

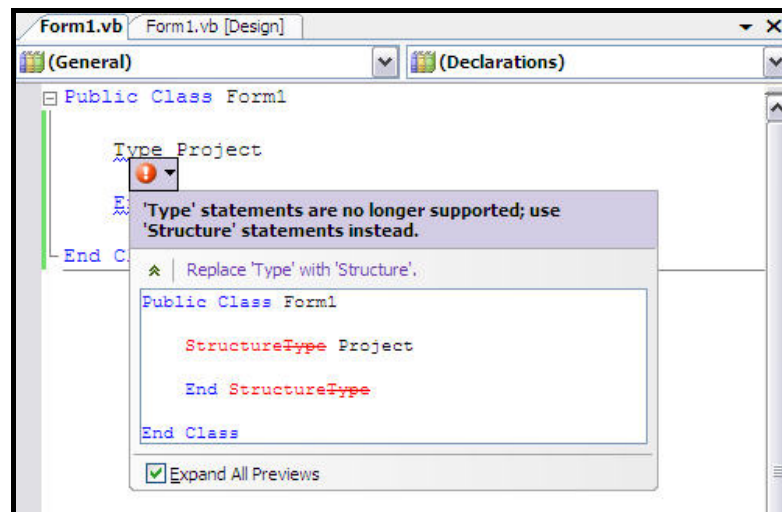


Diagram 4.11.6.2: Long description of a syntax error

- **Indentation** - If the code is not indented while it is being typed Visual Basic 2005 Code Editor will automatically indent the code



Diagram 4.11.7.1: Code written without indentation

Press enter and the indentation is done.

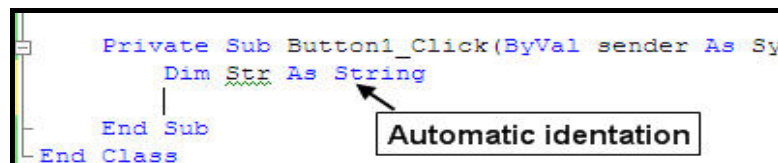




Diagram 4.11.7.2: Visual Basic 2005 automatically indented the code

- **Collapse Region** - One great feature of the code editor is the ability to collapse regions of code so that only the header can be seen. There is a  (plus sign) next to every method/function/class and clicking on them would expand the region and the sign changes to  (minus). Three dots appear near the header whose region is collapsed. If this feature needs to be turned off then select **Edit** → **Outlining** → **Stop Outlining** menu item. If this feature is needed (if it was switched off earlier) then select **Edit** → **Outlining** → **Collapse to Definitions** menu item

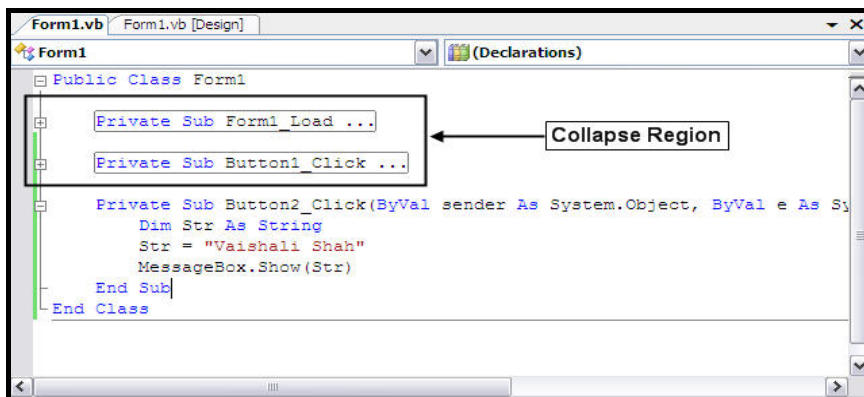


Diagram 4.11.8: Collapse Region

- ❑ **Code Snippets** - Visual Studio now provides segments of sample code ready to insert into a Visual Basic project. To display a list of available code snippets, right-click the active document in the Code Editor and select **Insert Snippet** from the context menu. Select the name of the snippet needed, the code is inserted into the editor and then it can be modified as desired

To manage the folders in which code snippets are stored and to add new snippets, select **Code Snippet Manager** from the Tools menu

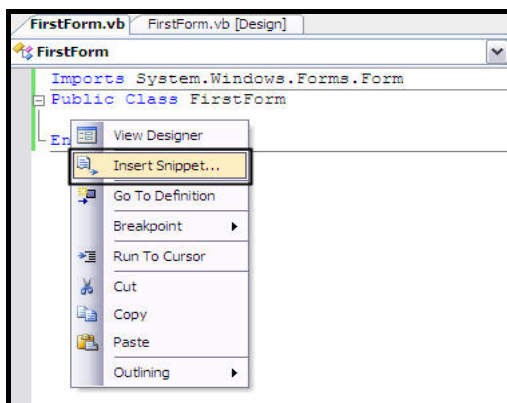


Diagram 4.11.9.1: Selecting Insert Snippet

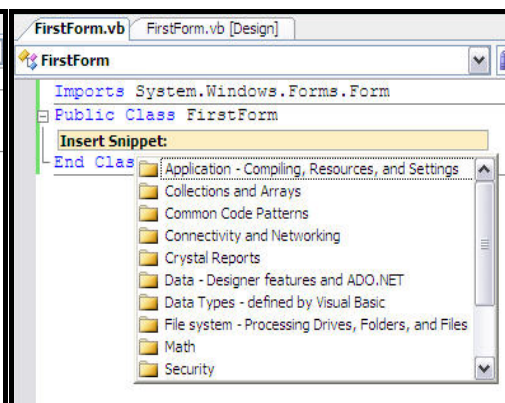


Diagram 4.11.9.2: Selecting the name of snippet

The Code Editor and the Form Designer provide two different ways to work with the same Visual Basic 2005 source file. The Code Editor allows working directly with the statements that make up the application. The Form Designer presents a visual representation of the forms and controls that are implemented by that code.

4. GETTING STARTED WITH THE IDE

Adding Event Handlers / Methods

When a component is double clicked, Visual Basic 2005 chooses one of its events to display the code block for. The chosen event represents Visual Basic 2005's guess as to which event is needed based on statistics. For example, the most common behavior with a Button is that the user clicks it. So when a code editor window is opened by double clicking a Button, Visual Basic 2005 shows that button's Click event. If another event or method is needed, first select the component or the control whose event is needed as shown in diagram 4.12.1 and then select the required event from Event / Method Name drop down list as shown in diagram 4.12.2.

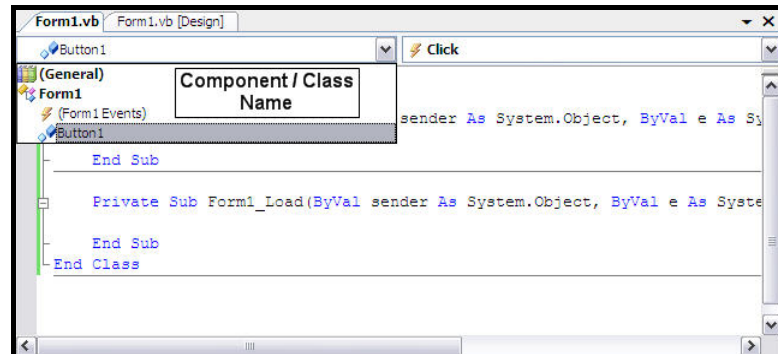


Diagram 4.12.1: Component / Class Button1 selected

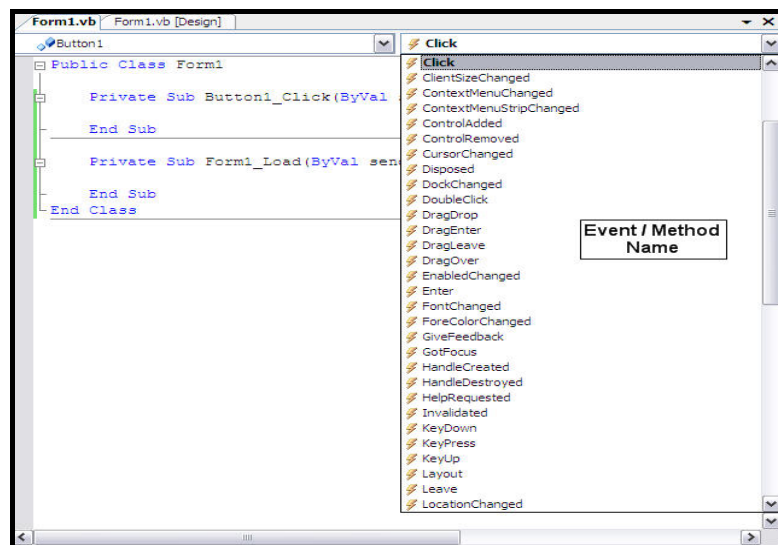



Diagram 4.12.2: Event / Method drop down list

Running The Project

To check whether the project is running successfully, click  or press **F5**. If it runs successfully (without any errors) the form will be presented as shown in diagram 4.13.

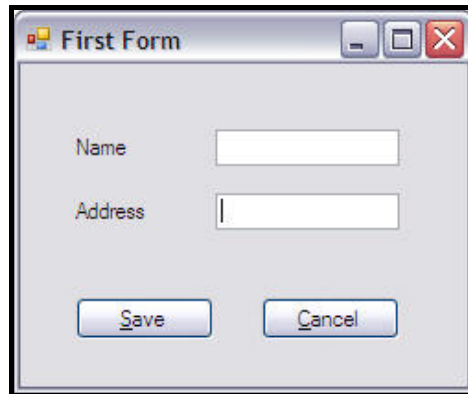


Diagram 4.13: First Form at runtime