Topic: 4.3.4 Use of development tools / programming environments

Features in editors that benefit programming:

VB.net Specific Basic Text Editor Features,

Visual Studio 2010

The **VB Specific** property page, in the **Basic** folder of the **Text Editor** folder of the **Options** (**Tools** menu) dialog box contains the following properties:

Automatic insertion of end constructs
When you type—for example, the first line of a procedure declaration, Sub Main—and press ENTER, the text editor adds a matching End Sub line. Similarly, if you add a For loop, the text editor adds a matching Next statement. When this option is selected, the code editor automatically adds the end construct.

Pretty Listing (reformatting) of code
The text editor reformats your code as appropriate. When this option is selected, the code editor will:

- Align your code to the correct tab position
- Recase keywords, variables, and objects to the correct case
- Add a missing **Then** to an **If...Then** statement
- Add parenthesis to function calls
- Add missing end quotes to strings
- Reformat exponential notation
- Reformat dates

Enable outlining mode
When you open a file in the code editor, you can view the document in outlining mode. See How to: Outline and Hide Code for more information. When this option is selected, the outlining feature is activated when you open a file.

Automatic insertion of Interface and MustOverride members
When you commit an **Implements** statement or an **Inherits** statement for a class, the text editor inserts prototypes for the members that have to be implemented or overridden, respectively.

Topic: 4.3.4 Use of development tools / programming environments

Show procedure line separators

The text editor indicates visual scope of procedures. A line is drawn in the .vb source files of your project at locations listed in the following table:

Location in .vb Source File	Example of Line Location
After the close of a block declaration construct	At the end of a class, structure, module, interface, or enum After a property, function, or sub Not between the get and set clauses in a property
After a set of single line constructs	After the import statements, before a type definition in a class file After variables declared in a class, before any procedures
After single line declarations (non-block level declarations)	Following import statements, inherits statements, variable declarations, event declarations, delegate declarations, and DLL declare statements

Enable error correction suggestions

The text editor can suggest solutions to common errors and allow you to select the appropriate correction, which is then applied to your code.

Enable highlighting of references and keywords

The text editor can highlight all instances of a symbol or all of the keywords in a clause such as **If..Then**, **While...End While**, or**Try...Catch...Finally**. You can navigate between highlighted references or keywords by pressing CTRL+SHIFT+DOWN ARROW or CTRL+SHIFT+UP ARROW.

Topic: 4.3.4 Use of development tools / programming environments

Debugging techniques

There are many like Breakpoints, DataTips and Watch Windows Debugging:

Introduction

In the software development life cycle, testing and defect fixing take more time than actually code writing. In general, debugging is a process of finding out defects in the program and fixing them. Defect fixing comes after the debugging, or you can say they are co-related. When you have some defects in your code, first of all you need to identify the root cause of the defect, which is called the debugging. When you have the root cause, you can fix the defect to make the program behavior as expected.

How to Start?

You can start debugging from the Debug menu of VS IDE. From the Debug Menu, you can select "Start Debugging" or just press F5 to start the program. If you have placed breakpoints in your code, then execution will begin automatically.

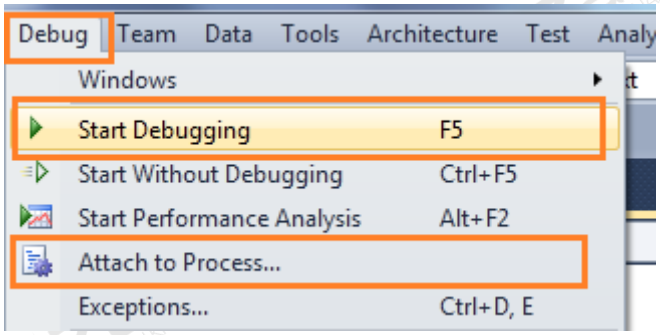

Figure: Start Debugging

We generally start debugging any application just by putting breakpoint on code where we think the problem may occur. So, let's start with breakpoints.

Breakpoints

Breakpoint is used to notify debugger where and when to pause the execution of program. You can put a breakpoint in code by clicking on the side bar of code or by just pressing F9 at the front of the line. So before keeping a breakpoint, you should know what is going wrong in your code and where it has to be stopped. When the debugger reaches the breakpoint, you can check out what's going wrong within the code by using a different debugging tool.

Topic: 4.3.4 Use of development tools / programming environments

Figure: Set Breakpoint

Debugging with Breakpoints

You have already set a breakpoint in your code where you want to pause the execution. And now start the program by pressing **"F5"**. When the program reaches the breakpoint, execution will automatically pause. Now you have several options to check your code. After hitting the breakpoint, breakpoint line will show as yellow color which indicates that this is the line which will execute next.

Now you have several commands available in break mode, using which you can proceed for further debugging.

Figure: Breakpoint Toolbar

Step Over

After debugger hits the breakpoint, you may need to execute the code line by line. **"Step Over" [F10]** command is used to execute the code line by line. This will execute the currently highlighted line and then pause. If you select F10 while a method call statement is highlighted, the execution will stop after the next line of the calling statement. Step Over will execute the entire method at a time.

Topic: 4.3.4 Use of development tools / programming environments

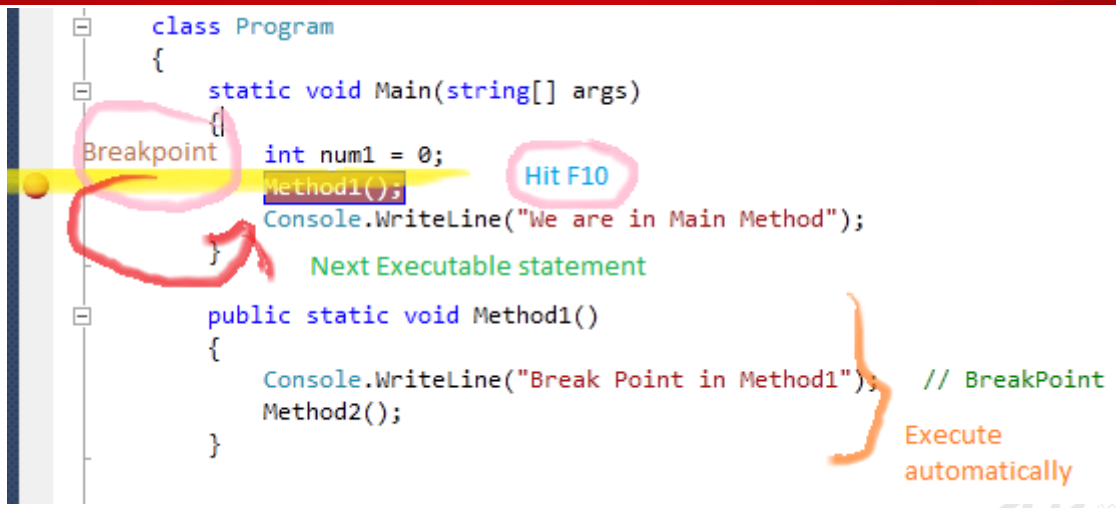

Figure: Step Over - F10

Step Into

This is similar to Step Over. The only difference is, if the current highlighted section is any methods call, the debugger will go inside the method. Shortcut key for Step Into is **"F11"**.

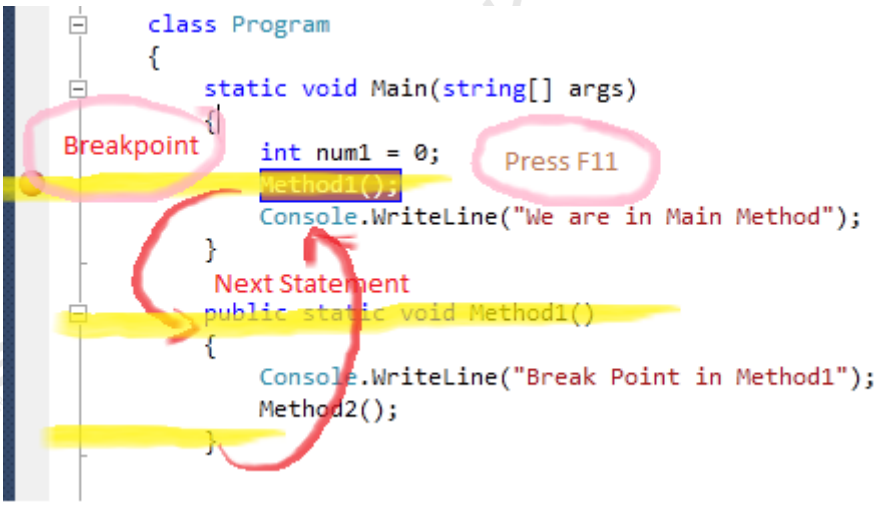

Figure: Step Into - F11

Step Out

This is related when you are debugging inside a method. If you press the **Shift - F11** within the current method, then the execution will complete the execution of the method and will pause at the next statement from where it called.

Topic: 4.3.4 Use of development tools / programming environments

Continue

It's like run your application again. It will continue the program flow unless it reaches the next breakpoint. The shortcut key for continue is **"F5"**.

Data Tip

Data tip is kind of an advanced tool tip message which is used to inspect the objects or variable during the debugging of the application. When debugger hits the breakpoint, if you mouse over to any of the objects or variables, you can see their current values. Even you can get the details of some complex object like dataset, datatable, etc. There is a **"+"** sign associated with the dataTip which is used to expand its child objects or variables.

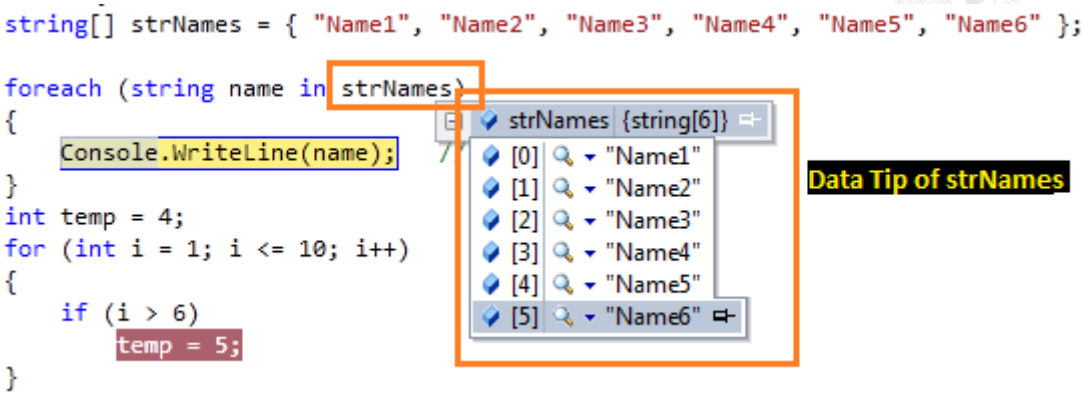

Figure: DataTips During Debugging

Watch Windows

You can say it is an investigation window. After breakpoint has been hit, the next thing you want to do is to investigate the current object and variables values. When you mouse hover on the variable, it shows the information as a data tip which you can expand, pin, import which I have already explained. There are various types of watch windows like Autos, Local, etc. Let's have a look into their details.

Topic: 4.3.4 Use of development tools / programming environments

Locals

It automatically displays the list of variables within the scope of current methods. If your debugger currently hits a particular breakpoint and if you open the "Autos" window, it will show you the current scope object variable along with the value.

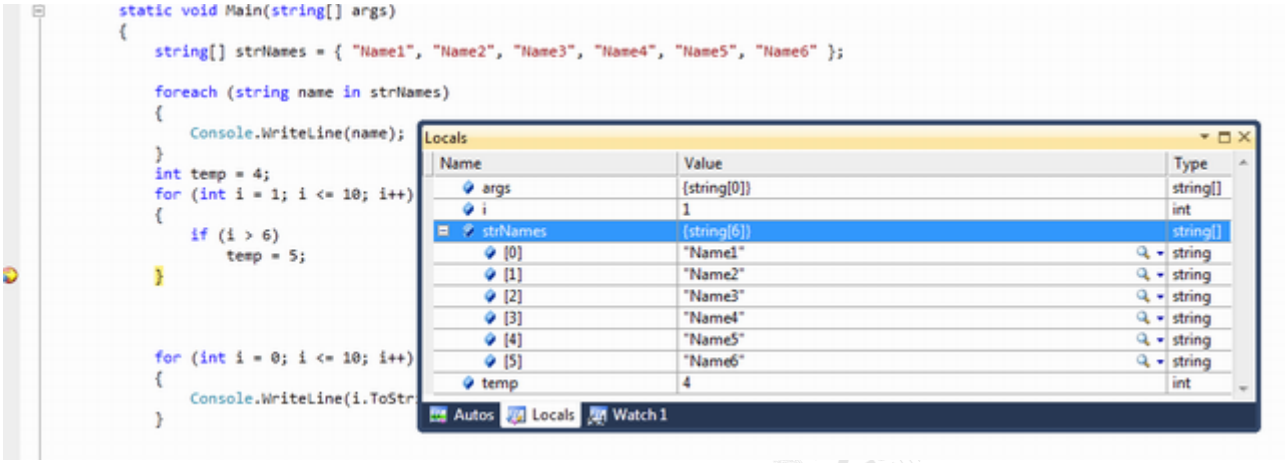

Figure: Local Variables

Autos

These variables are automatically detect by the VS debugger during the debugging. Visual Studio determines which objects or variables are important for the current code statement and based on that, it lists down the **"Autos"** variable. Shortcut key for the Autos Variable is **"Ctrl + D + A"**.

Figure: Autos - Ctrl+D, A

Topic: 4.3.4 Use of development tools / programming environments

Watch

Watch windows are used for adding variables as per requirement. It displays variables that you have added. You can add as many variables as you want into the watch window. To add variables in the watch window, you need to **"Right Click"** on variable and then select **"Add To Watch"**.

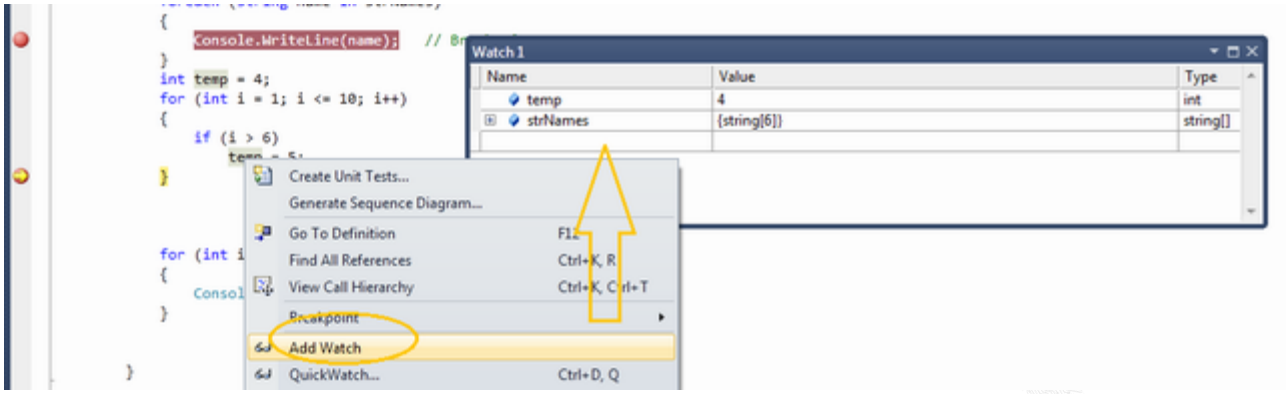

Figure: Watch - Ctrl+D, W

You can also use **Drag and Drop** to add variables in watch windows. If you want to delete any variable from watch window, just right click on that variable and select **"Delete Watch"**. From the debug window, you can also edit the variable value at run time.

Immediate Window

Immediate window is very much common and a favorite with all developers. It's very much helpful in debug mode of the application if you want to change the variable values or execute some statement without impacting your current debugging steps. You can open the Immediate window from menu **Debug** > **Window** >**Immediate Window** { **Ctrl + D, I** / **Alt + Ctrl - I** }. Immediate window has a set of commands which can be executed any time during debugging. During Debug mode, you can execute any command or execute any code statement from here.

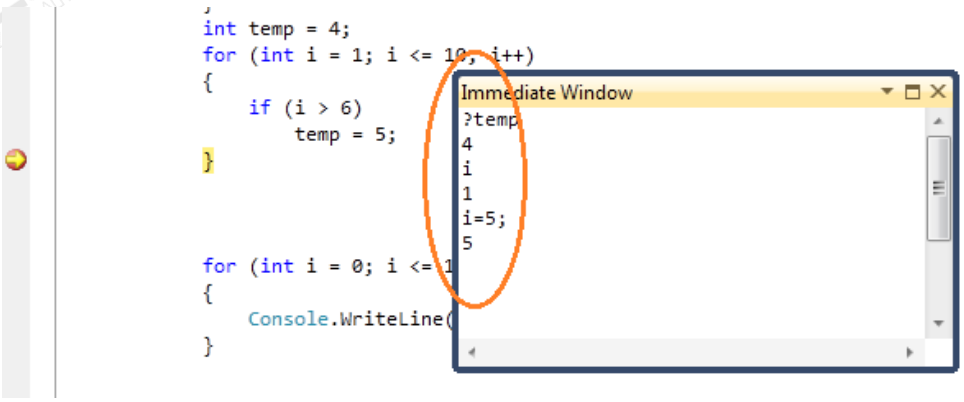

Figure: Basic Immediate Window

Topic: 4.3.4 Use of development tools / programming environments

Useful shortcut keys for visual studio Debugging

Shortcut Keys	Descriptions
Ctrl-Alt-V, A	Displays the Auto window
Ctrl-Alt-B	Displays the Breakpoints dialog
Ctrl-Shift-F9	Clears all of the breakpoints in the project
Ctrl-F9	Enables or disables the breakpoint on the current line of code
Ctrl-Alt-I	Displays the Immediate window
Ctrl-Alt-V, L	Displays the Locals window
Ctrl-Alt-Q	Displays the Quick Watch dialog
Ctrl-Shift-F5	Terminates the current debugging session, rebuilds if necessary, and starts a new debugging session.
Ctrl-F10	Starts or resumes execution of your code and then halts execution when it reaches the selected statement.
Ctrl-Shift-F10	Sets the execution point to the line of code you choose
F5	If not currently debugging, this runs the startup project or projects and attaches the debugger.
Ctrl-F5	Runs the code without invoking the debugger
F11	Step Into
Shift-F11	Executes the remaining lines out from procedure
F10	Executes the next line of code but does not step into any function calls
Shift-F5	Available in break and run modes, this terminates the debugging session
F9	Sets or removes a breakpoint at the current line

Topic: 4.3.4 Use of development tools / programming environments

Know when to use compilers and interpreters:

Compiler is used when the whole program is executed (black box testing). This is fast execution. Whereas interpreter is used when program is executed line by line (White box testing). In modern times, languages like VB.net uses both, i.e. interpreter during editing a program to rectify syntax errors then and there and compiler to run/execute/debug the program as a whole.

Comparison	
COMPILER	**INTERPRETER**
Fast, creates executable file that runs directly on the CPU	Slower, interprets code one line at a time
Debugging is more difficult. One error can produce many false errors	Debugging is easier. Each line of code is analysed and checked before being executed
More likely to crash the computer. The machine code is running directly on the CPU	Less likely to crash as the instructions are being carried out either on the interpreters command line or within a virtual machine environment which is protecting the computer from being directly accessed by the code.
Easier to protect Intellectual Property as the machine code is difficult to understand	Weaker Intellectual property as the source code has to be available at run time.
Uses more memory - all the execution code needs to be loaded into memory.	Uses less memory, source code only has to be present one line at a time in memory
Unauthorised modification to the code more difficult. The executable is in the form of machine code. So it is difficult to understand program flow	Easier to modify as the instructions are at a high level and so the program flow is easier to understand and modify.

Topic: 4.3.4 Use of development tools / programming environments

Software development methodology

Agile Software Development Methodology

Agile software development is a conceptual framework for undertaking software engineering projects. There are a number of agile software development methodologies e.g. Crystal Methods, Dynamic Systems Development Model (DSDM), and Scrum.

Most agile methods attempt to minimize risk by developing software in short time boxes, called iterations, which typically last one to four weeks. Each iteration is like a miniature software project of its own, and includes all the tasks necessary to release the mini-increment of new functionality: planning, requirements analysis, design, coding, testing, and documentation. While iteration may not add enough functionality to warrant releasing the product, an agile software project intends to be capable of releasing new software at the end of every iteration. At the end of each iteration, the team reevaluates project priorities.

Agile methods emphasize real time communication, preferably face-to-face, over written documents. Most agile teams are located in a bullpen and include all the people necessary to finish the software. At a minimum, this includes programmers and the people who define the product such as product managers, business analysts, or actual customers. The bullpen may also include testers, interface designers, technical writers, and management.

Agile methods also emphasize working software as the primary measure of progress. Combined with the preference for face-to-face communication, agile methods produce very little written documentation relative to other methods.

Rapid application development (RAD)

"Rapid-development language" is a general term that refers to any programming language that offers speedier implementation than do traditional third-generation languages such as C/C++, Pascal, or Fortran. Rapid-Development Languages (RDLs) produce their savings by reducing the amount of construction needed to build a product. Although the savings are realized during construction, the ability to shorten the construction cycle has project wide implications: shorter construction cycles make incremental lifecycles such as Evolutionary Prototyping practical. Because RDLs often lack first-rate performance, constrain flexibility, and are limited to specific kinds of problems, they are usually better suited to the development of in-house business software and limited-distribution custom software than systems software.

Topic: 4.3.4 Use of development tools / programming environments

RAD (rapid application development) proposes that products can be developed faster and of higher quality by:

- Using workshops or focus groups to gather requirements.
- Prototyping and user testing of designs.
- Re-using software components.
- Following a schedule that defers design improvements to the next product version.
- Keeping review meetings and other team communication informal.

There are commercial products that include requirements gathering tools, prototyping tools, software development environments such as those for the Java platform, groupware for communication among development members, and testing tools. RAD usually embraces object-oriented programming methodology, which inherently fosters software re-use. The most popular object-oriented programming languages, C++ and Java, are offered in visual programming packages often described as providing rapid application development.

Waterfall model

The waterfall model is a popular version of the systems development life cycle model for software engineering. Often considered the classic approach to the systems development life cycle, the waterfall model describes a development method that is rigid and linear. Waterfall development has distinct goals for each phase of development where each phase is completed for the next one is started and there is no turning back.

The perceived advantages of the waterfall process are that it allows for departmentalization and managerial control. A schedule is typically set with deadlines for each stage of development and a product can proceed through the development process. In theory, this process leads to the project being delivered on time because each phase has been planned in detail.

In practice, waterfall development often falls short of expectations as it does not embrace the inevitable changes and revisions that become necessary with most projects. Once an application is in the testing stage, it is very difficult to go back and change something that was not thought of in the concept stage.