Dnyansagar Arts and commerece college Balewadi pune

BBA(CA) V SEM- 2013 (PATTERN)

DOT NET PROGRAMMING-(503)

By
Prof Gayatri A Amate

Unit 1

INTRODUCTION TO .NET FRAMEWORK

.NET – WHAT IS IT?

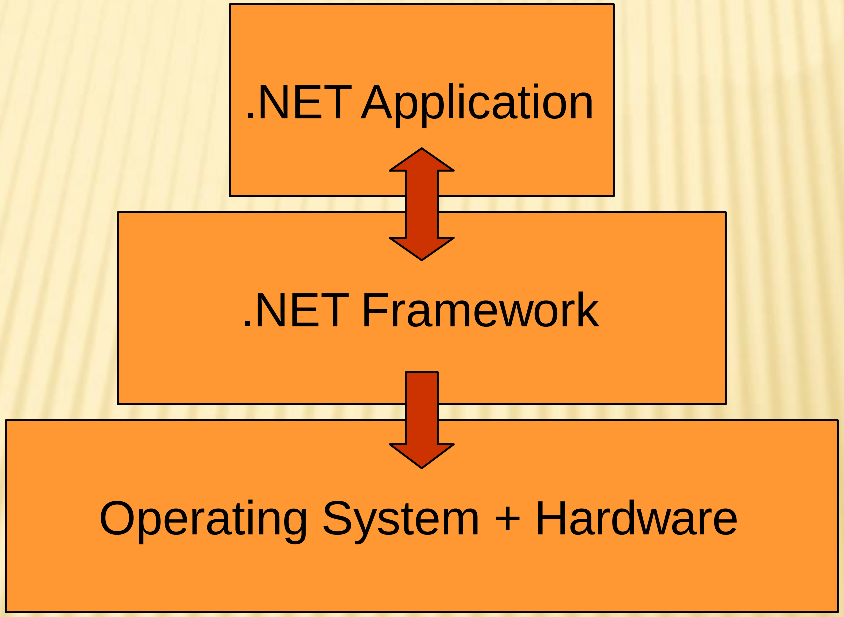
- Software platform
- Language neutral
- In other words:

  .NET is not a language (Runtime and a library for writing and executing written programs in any compliant language)

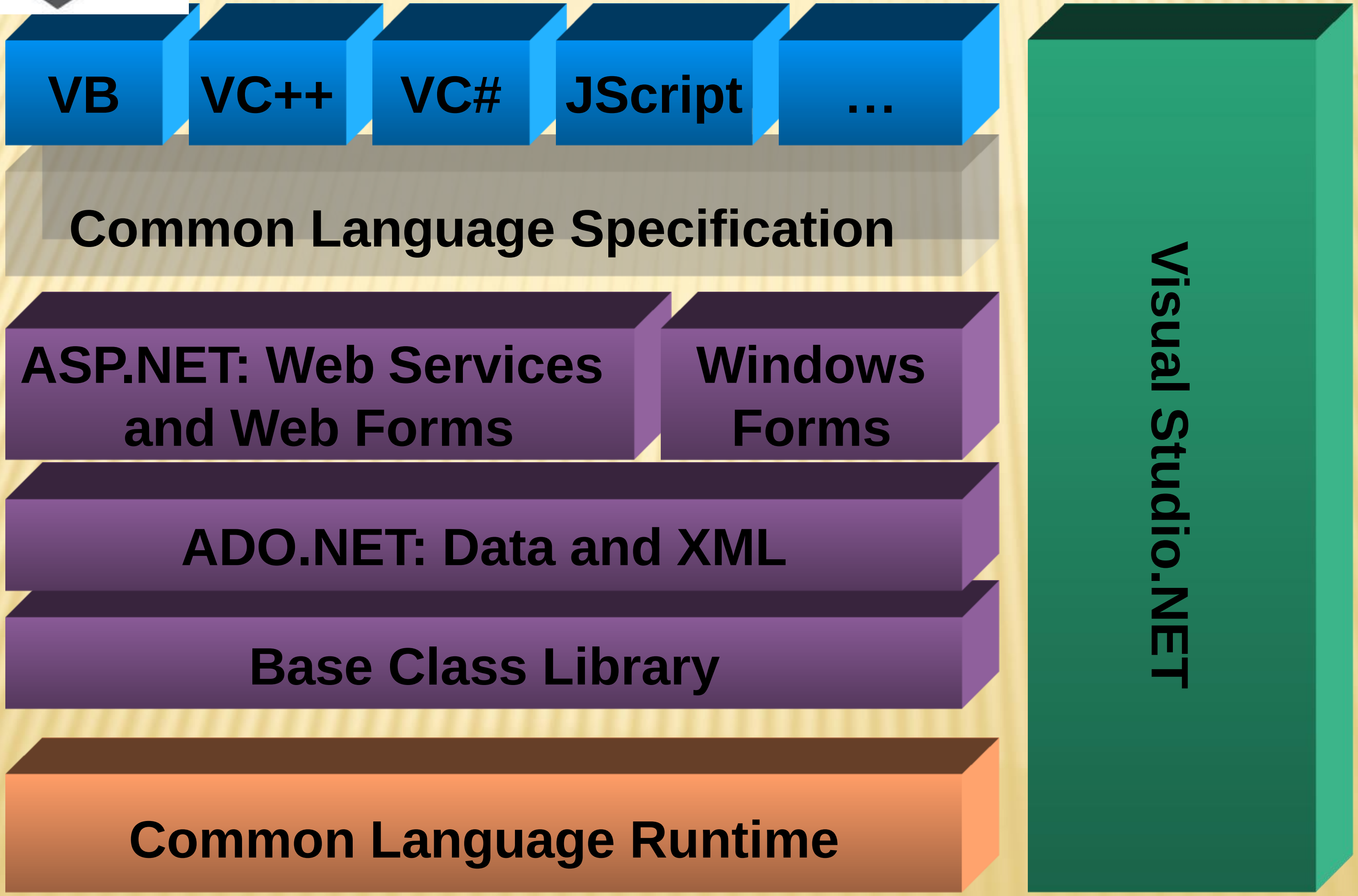
WHAT IS .NET

- .Net is a new framework for developing web-based and windows-based applications within the Microsoft environment.
- The framework offers a fundamental shift in Microsoft strategy: it moves application development from client-centric to server-centric.

.NET – WHAT IS IT?



FRAMEWORK, LANGUAGES, AND TOOLS



THE .NET FRAMEWORK

.NET Framework Services

- Common Language Runtime
- Windows® Forms
- ASP.NET
  - Web Forms
  - Web Services
- ADO.NET, evolution of ADO
- Visual Studio.NET

COMMON LANGUAGE RUNTIME (CLR)

•CLR works like a virtual machine in executing all languages.

•All .NET languages must obey the rules and standards imposed by CLR. Examples:

- Object declaration, creation and use
- Data types,language libraries
- Error and exception handling
- Interactive Development Environment (IDE)

COMMON LANGUAGE RUNTIME

- Common Language Specification (CLS)
Common Type System (CTS)
- Standard class framework Automatic
memory management

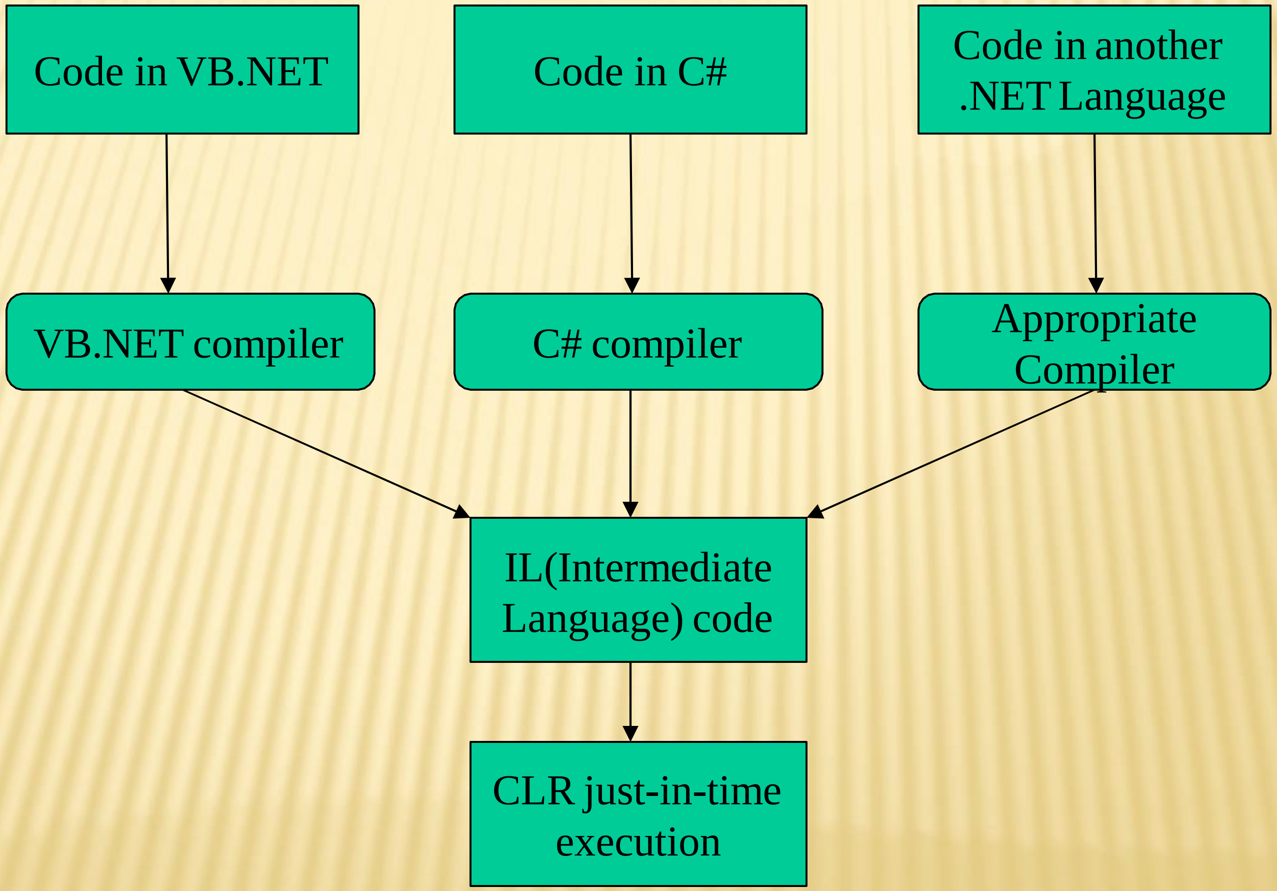
- Development
  - Mixed language applications
-
-
-
-
  - Consistent error handling and safer execution
  - Potentially multi-platform
- Deployment
  - Removal of registration dependency
  - Safety – fewer versioning problems

COMMON LANGUAGE RUNTIME

Multiple Language Support

- CTS is a rich type system built into the CLR
  - Implements various types (int, double, etc)
  - And operations on those types
- CLS is a set of specifications that language and library designers need to follow
  - This will ensure interoperability between languages

COMPILATION IN .NET


Code in VB.NET
Code in C#
Code in another .NET Language
VB.NET compiler
C# compiler
Appropriate Compiler
IL(Intermediate Language) code
CLR just-in-time execution

INTERMEDIATE LANGUAGE (IL)

- .NET languages are not compiled to machine code. They are compiled to an Intermediate Language (IL).

- CLR accepts the IL code and recompiles it to machine code. The recompilation is just-in-time (JIT) meaning it is done as soon as a function or subroutine is called.

- The JIT code stays in memory for subsequent calls. In cases where there is not enough memory it is discarded thus making JIT process interpretive.

LANGUAGES

- Languages provided by MS
  - VB, C++, C#, J#, JScript
- Third-parties are building
  - APL, COBOL, Pascal, Eiffel, Haskell, ML, Oberon, Perl, Python, Scheme, Smalltalk…

ASP.NET

- Logical Evolution of ASP
  - Supports multiple languages
  - Improved performance
  - Control-based, event-driven execution model
  - More productive
  - Cleanly encapsulated functionality

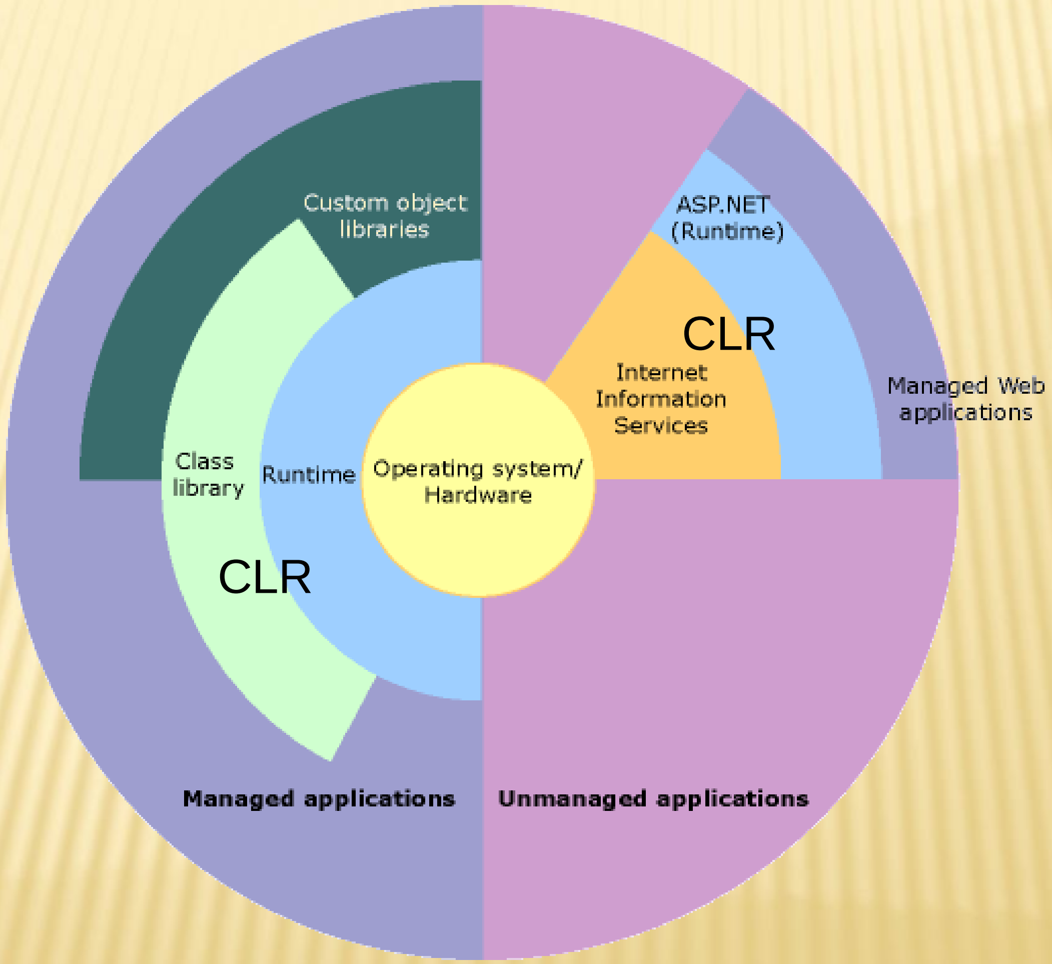
ADO.NET
(DATA AND XML)

- New objects (e.g., DataSets)
- Separates connected / disconnected issues
- Language neutral data access
- Uses same types as CLR
- Great support for XML

VISUAL STUDIO.NET

- Development tool that contains a rich set of productivity and debugging features

NET – HIERARCHY, ANOTHER VIEW



NET Tools

Visual Studio .NET is Microsoft's flagship tool for developing Windows software.

Visual Studio provides an integrated development environment (IDE) for developers to create standalone Windows applications, interactive Web sites, Web applications, and Web services running on any platform that supports .NET.

TOOL

In addition, there are many .NET Framework tools designed to help developers **create, configure, deploy, manage and secure .NET applications and components.**

SUMMARY

- The .NET Framework
  - Dramatically simplifies development and deployment
  - Provides robust and secure execution environment
  - Supports multiple programming languages

UNIT 2

Introduction to VB.net

Operators in VB.NET

- Arithmetic operators
- Comparison operators
- Logical operators
- Concatenation operators

Arithmetic operators

- Arithmetic operators are used to perform arithmetic calculations such as addition and subtraction.
- VB.NET supported arithmetic are listed in the given table.

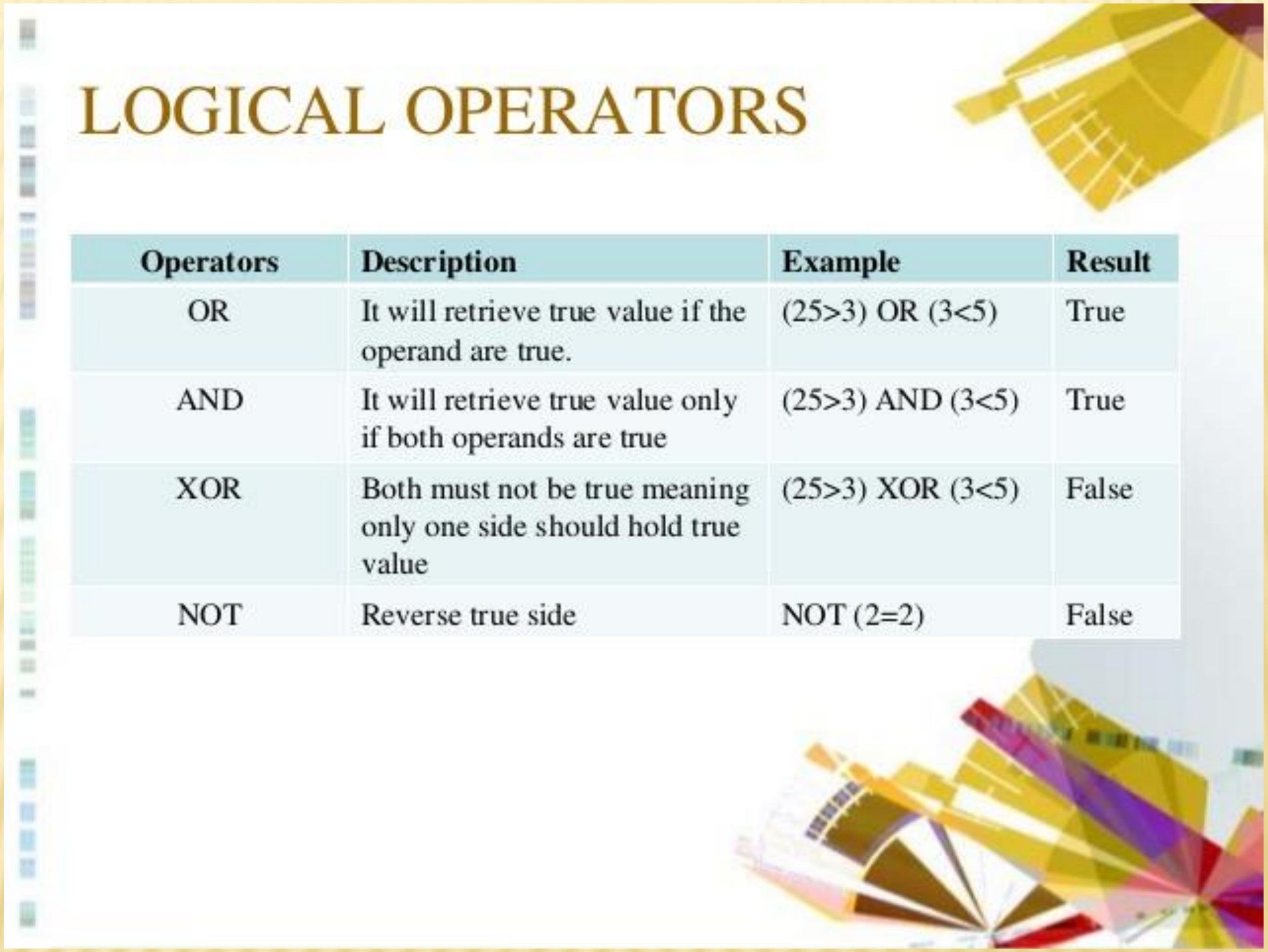
Operator	Purpose	Example: Let a=10	Output
+	Addition	b = a + 5	55
-	Subtraction	c = a – 5	45
*	Multiplication	d = a * 10	500
/	Division	e = a / 6	8.33
\	Division (integer part only given as output)	f = a \ 6	8
MOD	Modulas (Remainder after division)	g = a mod 7	1
^	Power	h = a^2	2500
=	Assignment	m = 100	

Comparison operators

- Comparison operators are used to compare two or more values.
- VB.NET supported comparison operators are listed in the following table.

Operator	Purpose	Example: Let a=50	Output
=	Check if a value is equal to another	Dim b As Boolean b=(a=55)	FALSE
<>	Check if a value is not equal to another	b=(a<>55)	TRUE
>	Check if a value is greater than another	b=(a>100)	FALSE
<	Check if a value is less than another	b=(a<100)	TRUE
>=	Check if a value is greater than or equal to another	b=(a>=50)	TRUE
<=	Check if a value is less than or equal to another	b=(a<=75)	TRUE
LIKE	Check if a string matches a given pattern	Dim P As string P="Good" b=("God" LIKE P)	FALSE
IS	Check if two object references refer to a same object.		

Logical operators



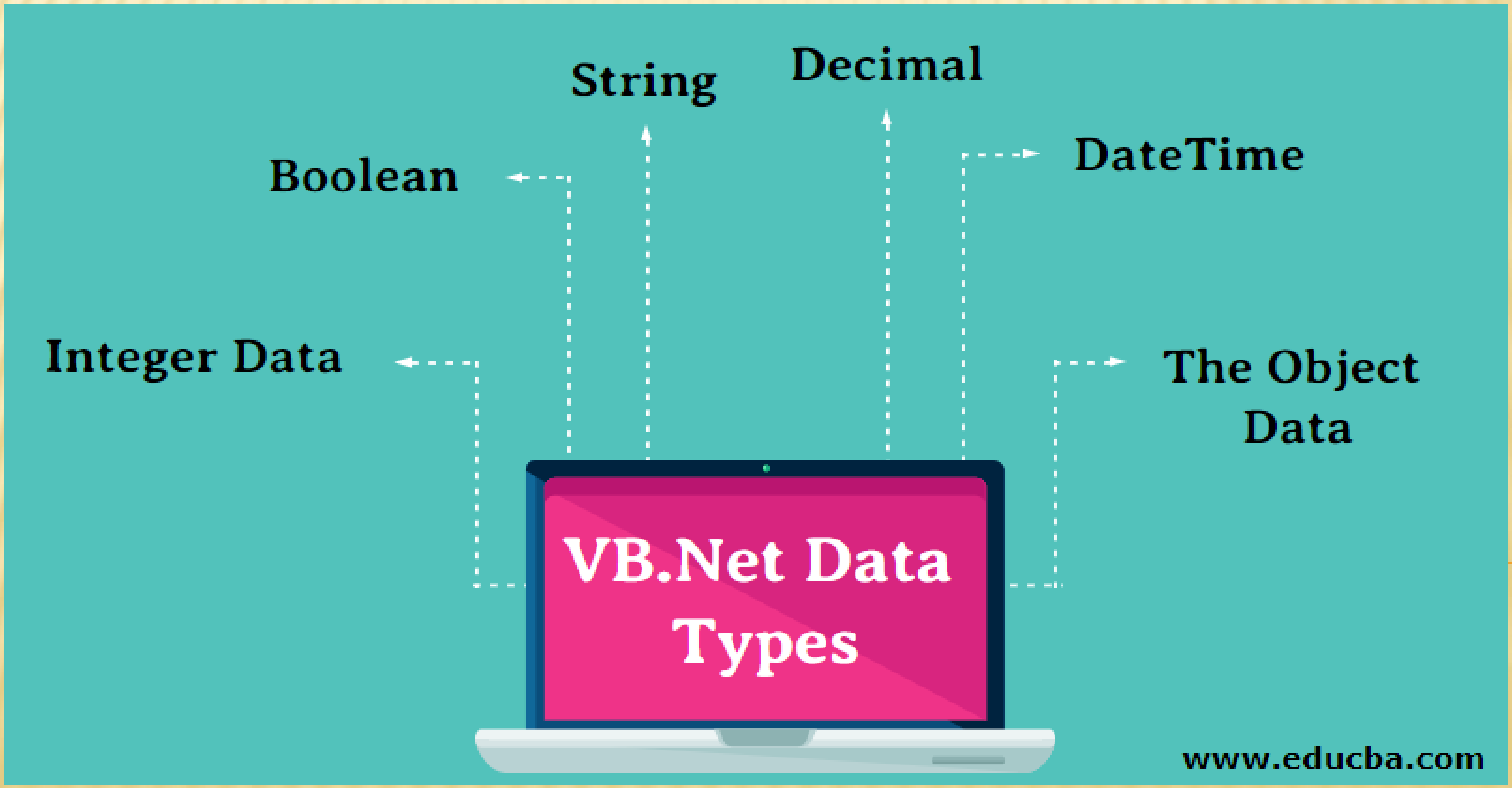
LOGICAL OPERATORS

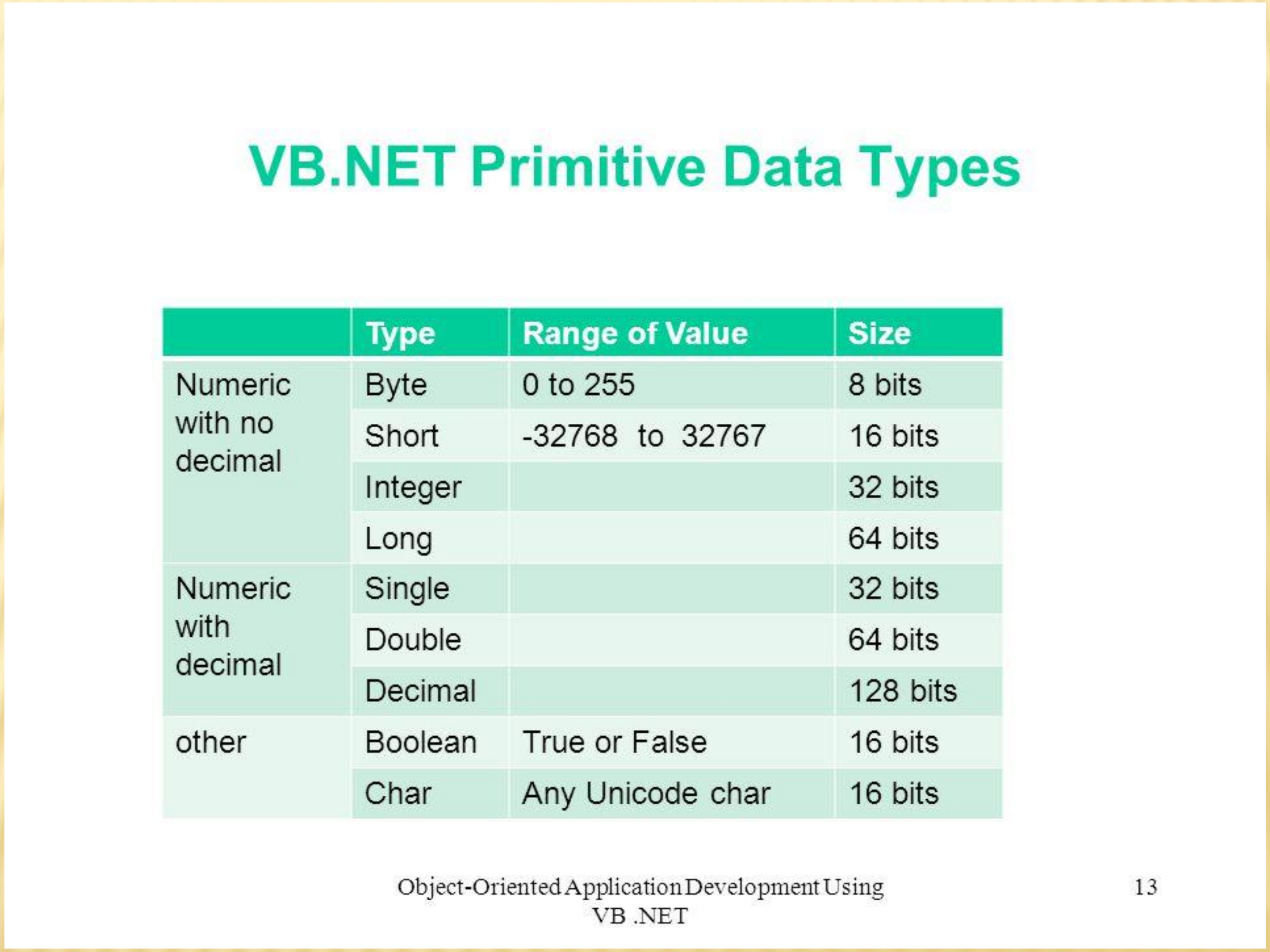
Operators	Description	Example	Result
OR	It will retrieve true value if the operand are true.	(25>3) OR (3<5)	True
AND	It will retrieve true value only if both operands are true	(25>3) AND (3<5)	True
XOR	Both must not be true meaning only one side should hold true value	(25>3) XOR (3<5)	False
NOT	Reverse true side	NOT (2=2)	False

DATA TYPES

- Data types refer to an extensive system used for declaring variables or functions of different types.
- The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

DATA TYPES AVAILABLE IN VB.NET





VB.NET Primitive Data Types

	Type	Range of Value	Size
Numeric with no decimal	Byte	0 to 255	8 bits
	Short	-32768 to 32767	16 bits
	Integer		32 bits
	Long		64 bits
Numeric with decimal	Single		32 bits
	Double		64 bits
	Decimal		128 bits
other	Boolean	True or False	16 bits
	Char	Any Unicode char	16 bits

Control Flow

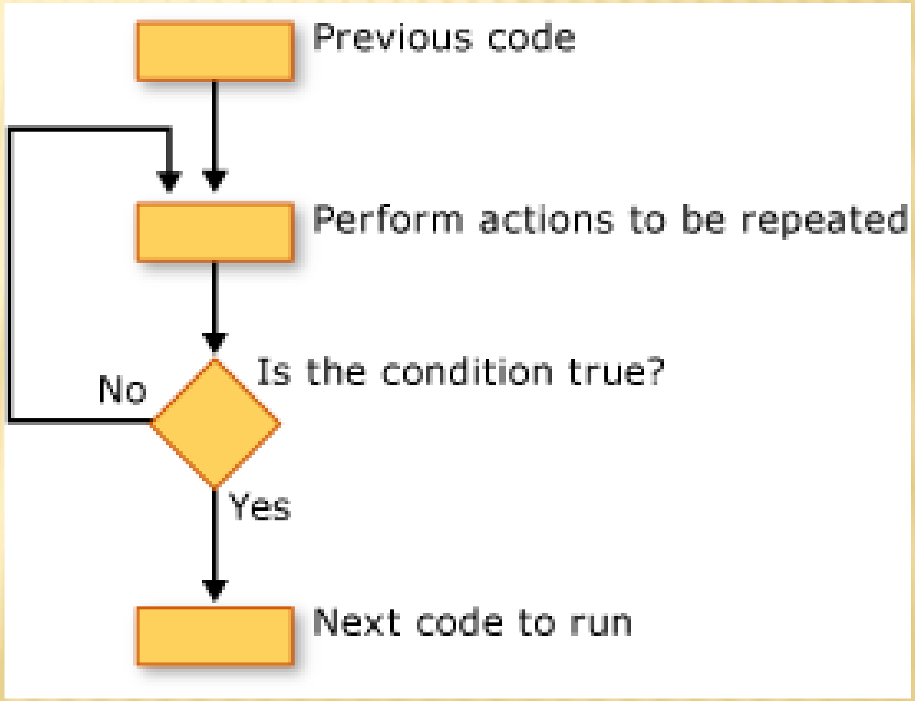
In a program, statements may be executed sequentially, selectively or iteratively. Every programming language provides constructs to support sequence, selection or iteration. So there are three types of programming constructs :

- **Sequence**
  - Functions and Procedures
- **Selection**
  - If...Then...Else statement
  - Select Case statement
- **Iterative**
  - For...Next Loop statement
  - Do...Loop statement

Loop Structure

There are mainly 3 types of loop:

- ➢For Loop
- ➢While Loop
- ➢Do Loop



Example

```
Module Module1

Sub Main()
Dim d As Integer
For d = 0 To 2
System.Console.WriteLine("In the For Loop")
Next d
End Sub

End Module
```

FOR Loop

- The For loop is the most popular loop.

- For loops enable us to execute a series of expressions multiple numbers of times.

- **Syntax:**

```
For index=start to end[Step step]
[statements]
[Exit For]
[statements]
Next[index]
```

While Loop

- While loop keeps executing until the condition against which it tests remain true.

- **Syntax:**

```
 While condition
[statements]
End While
```

```
Module Module1

Sub Main()
Dim d, e As Integer
d = 0
e =6
While e > 4
e -= 1
d += 1
End While
System.Console.WriteLine("The Loop ran " & e &
"times")
End Sub

End Module
```

DO Loop

- The Do loop keeps executing it's statements while or until the condition is true.
- Two keywords, while and until can be used with the do loop.
- The Do loop also supports an Exit Do statement which makes the loop to exit at any moment.

- **Syntax:**

```
Do[{while | Until} condition]
[statements]
[Exit Do]
[statements]
Loop
```

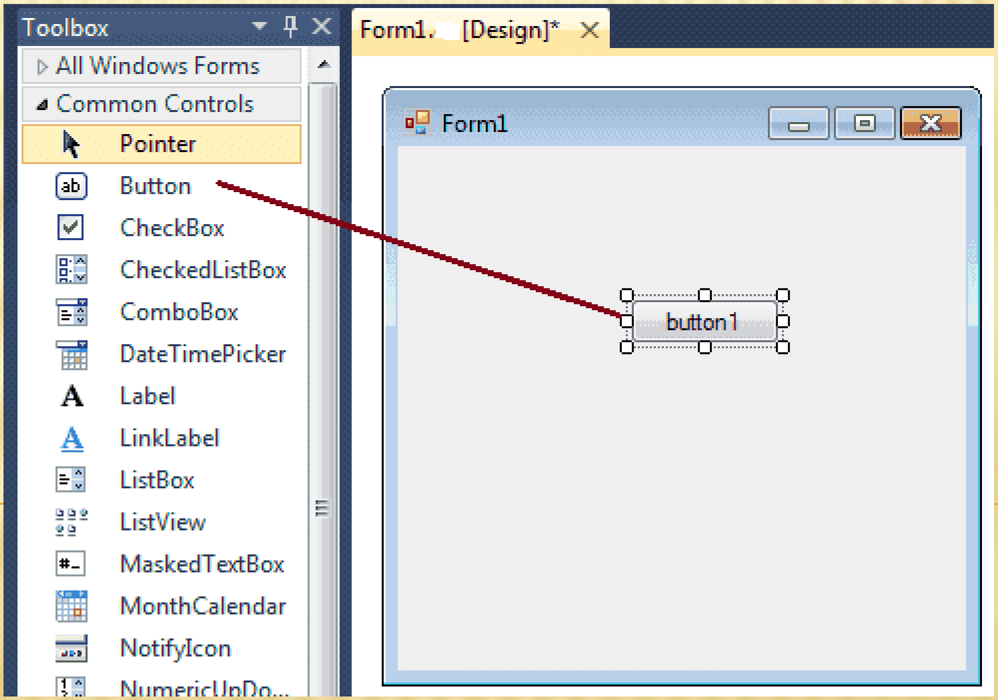
Example:

```
Module Module1

Sub Main()
Dim str As String
Do Until str = "Cool"
System.Console.WriteLine("What to do?")
str = System.Console.ReadLine()
Loop
End Sub

End Module
```

WINDOWS FORMS

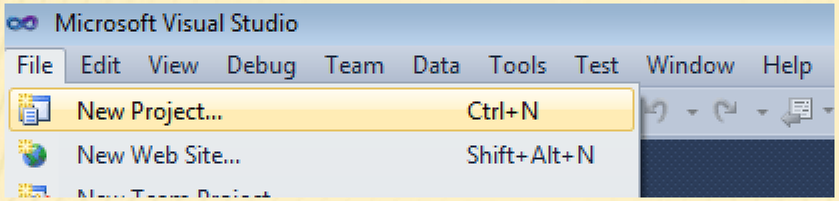
VB.Net programmers have made extensive use of forms to build user interfaces.

Each time you create a Windows application, Visual Studio will display a default blank form, onto which you can drag and drop controls from the Visual Studio Toolbox window

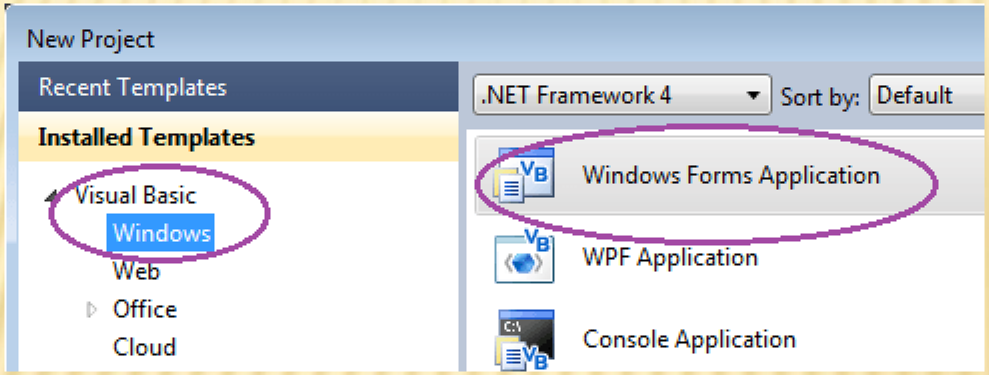


CREATE NEW FORM IN VB.NET

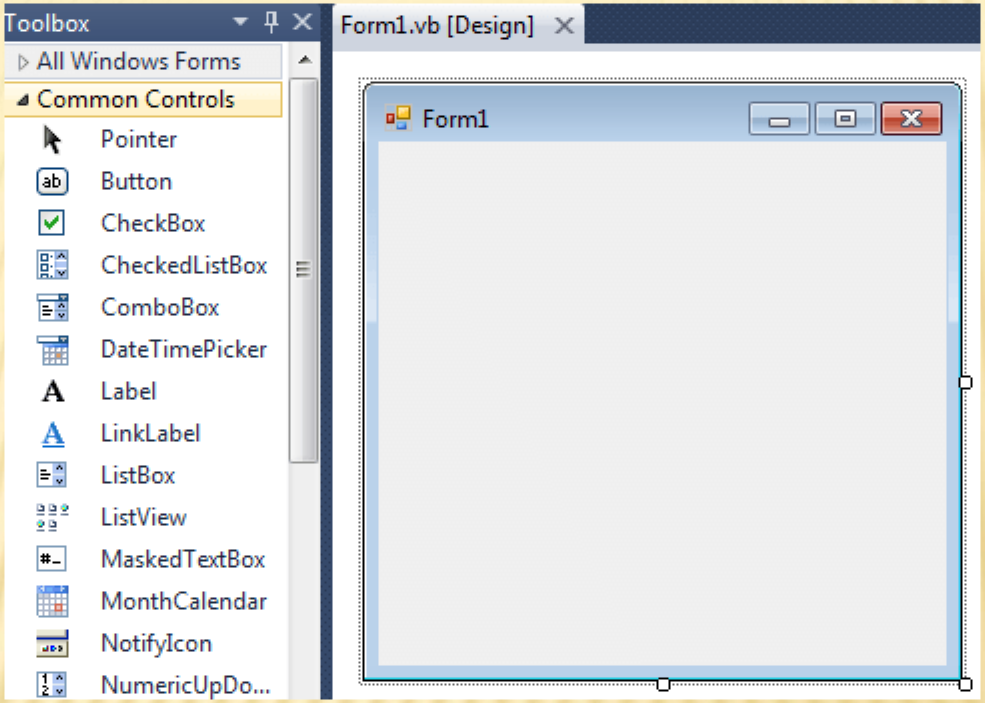
Step 1.


Microsoft Visual Studio
File Edit View Debug Team Data Tools Test Window Help
New Project... Ctrl+N
New Web Site... Shift+Alt+N

Step 2 : Select project type from New project dialog Box.

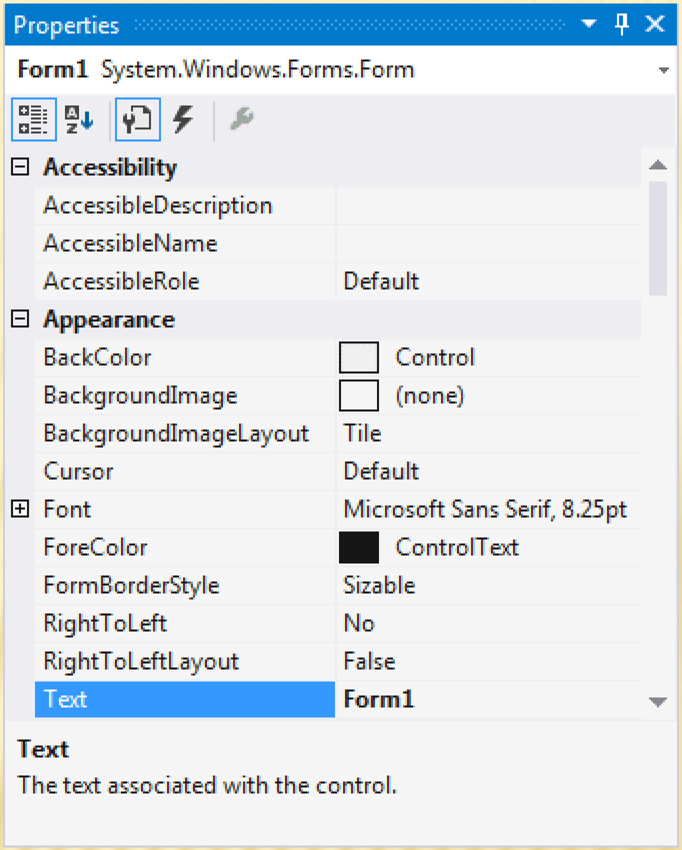


When you add a Windows Form to your project, many of the forms properties are set by default. Although these values are convenient, they will not always suit your programming needs. The following picture shows how is the default Form look like.



Form Properties

Following table lists down various important properties related to a form. These properties can be set or read during application execution. You can refer to Microsoft documentation for a complete list of properties associated with a Form control −



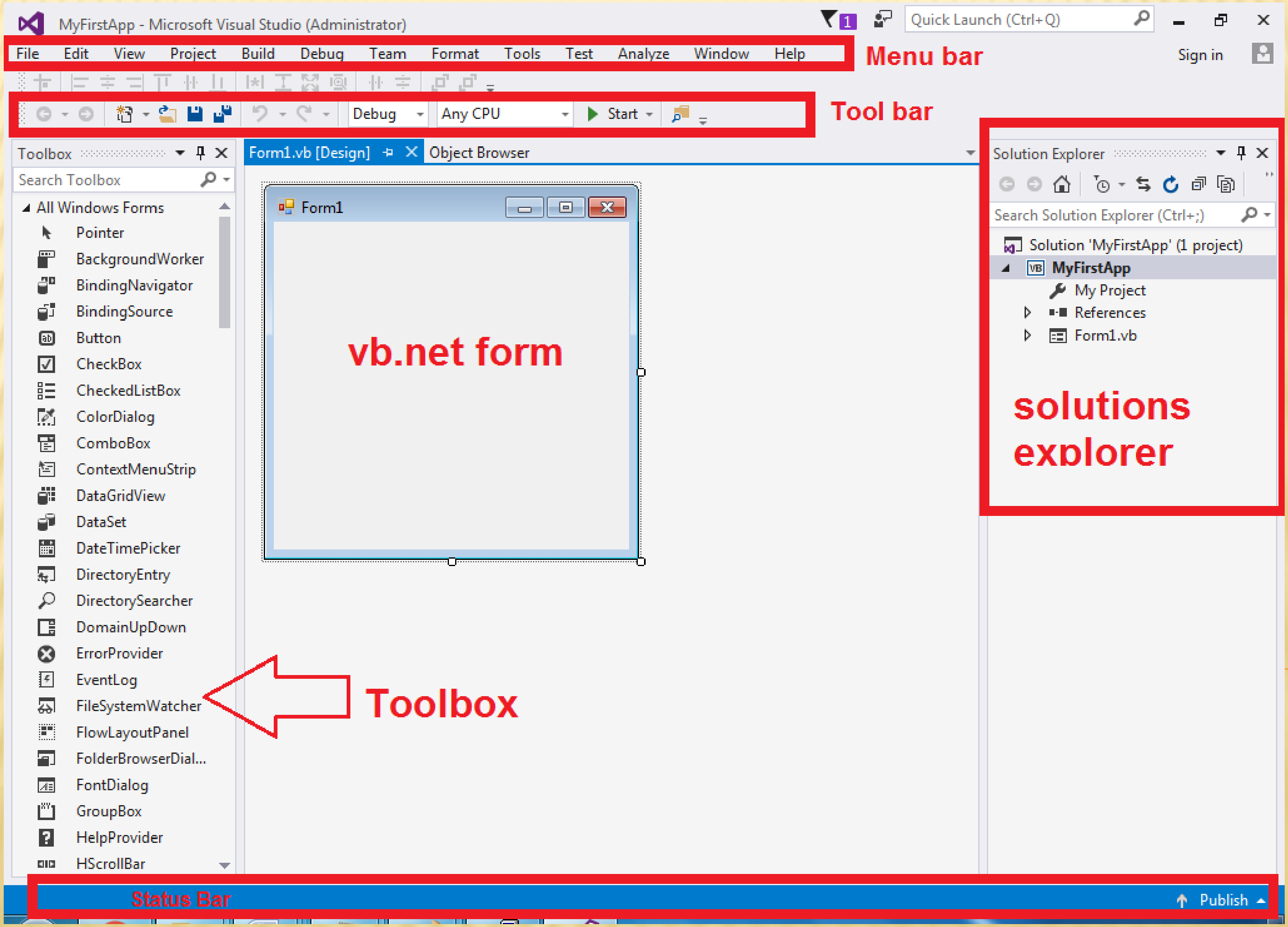
Form1 System.Windows.Forms.Form

⊟ **Accessibility**	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
⊟ **Appearance**	
BackColor	☐ Control
BackgroundImage	☐ (none)
BackgroundImageLayout	Tile
Cursor	Default
⊞ Font	Microsoft Sans Serif, 8.25pt
ForeColor	■ ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	**Form1**

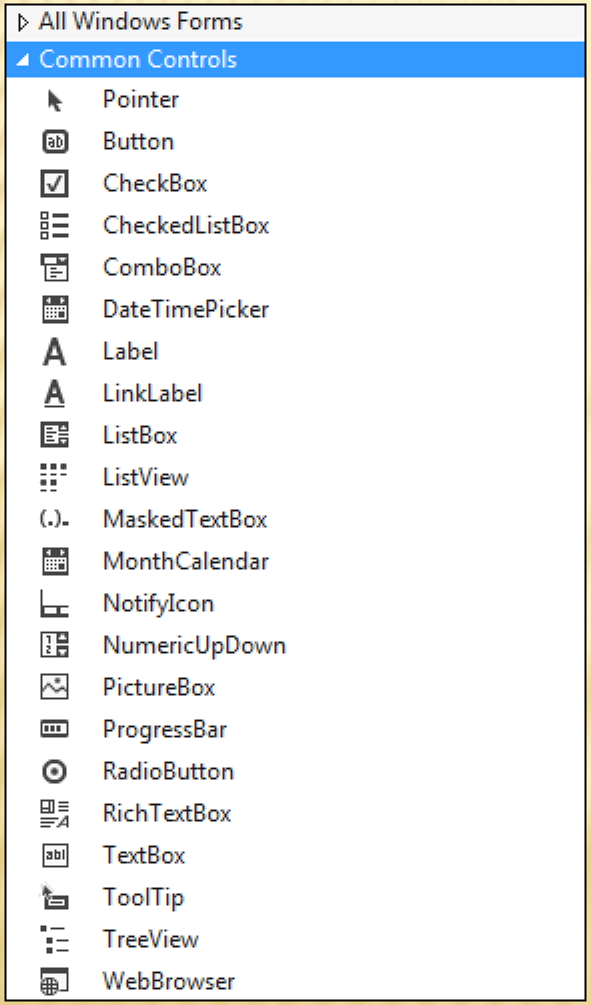
Text

The text associated with the control.

VISUAL STUDIO IDE



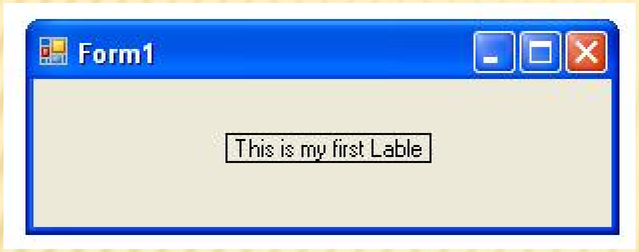
Visual Basic.NET IDE is built out of a collection of different windows. Some windows are used for writing code, some for designing interfaces, and others for getting a general overview of files or classes in your application.

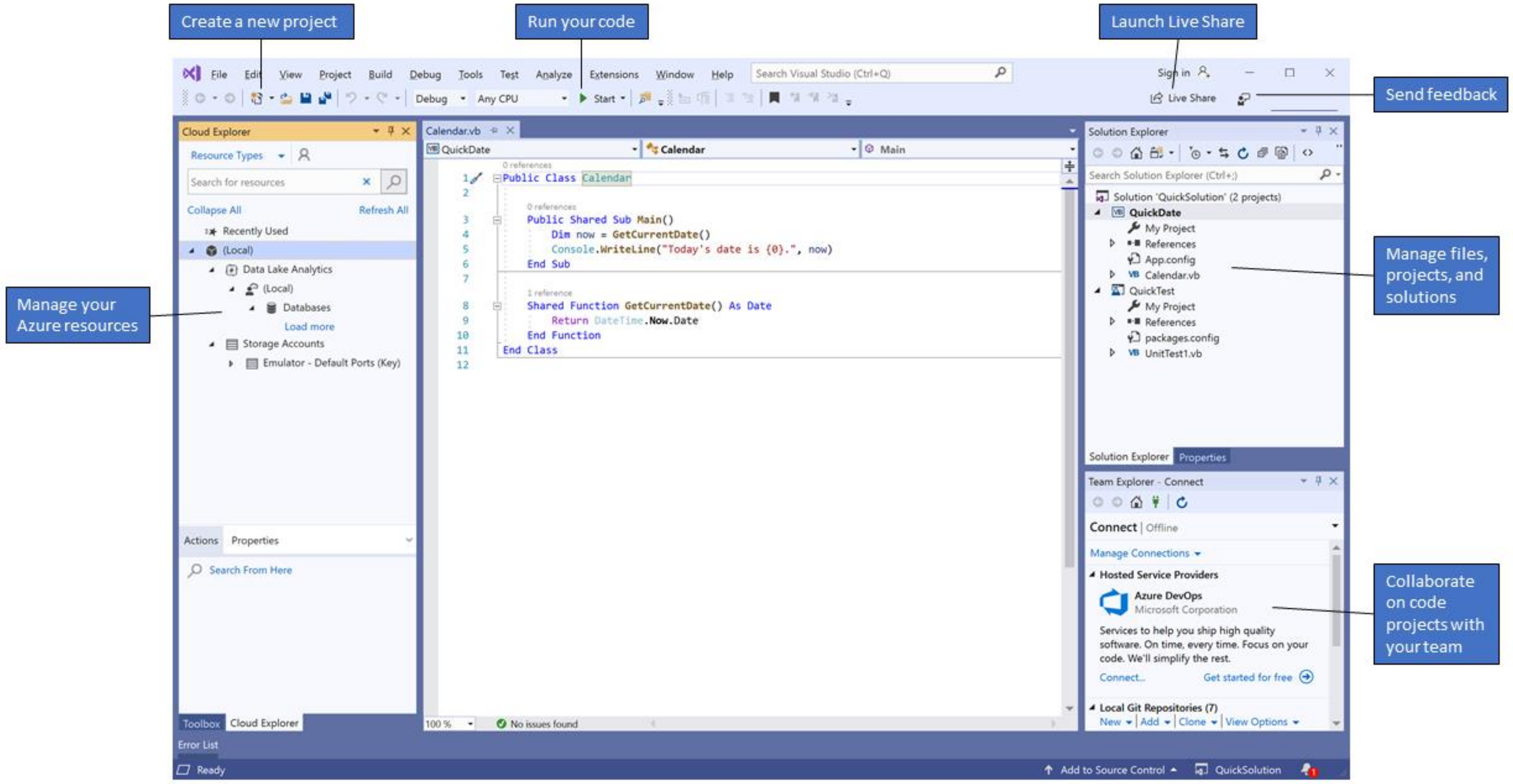


Label Control

Microsoft Visual Studio .NET controls are the graphical tools you use to build the user interface of a VB.Net program. Labels are one of the most frequently used Visual Basic control.

A Label control lets you place descriptive text , where the text does not need to be changed by the user. The Label class is defined in the System.Windows.Forms namespace.



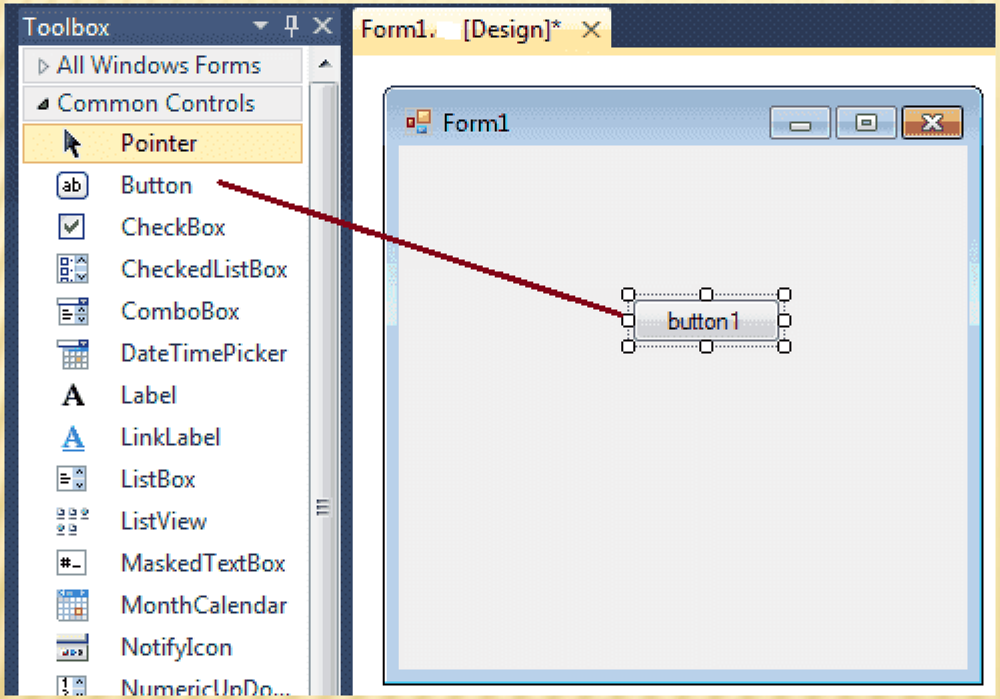


```
Public Class Calendar

    Public Shared Sub Main()
        Dim now = GetCurrentDate()
        Console.WriteLine("Today's date is {0}.", now)
    End Sub

    Shared Function GetCurrentDate() As Date
        Return DateTime.Now.Date
    End Function
End Class
```

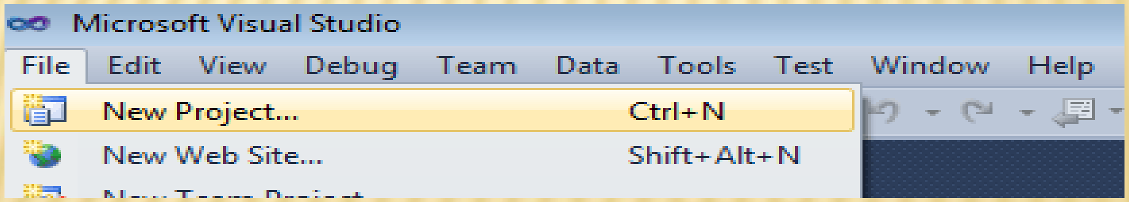
Services to help you ship high quality software. On time, every time. Focus on your code. We'll simplify the rest.

Windows Forms

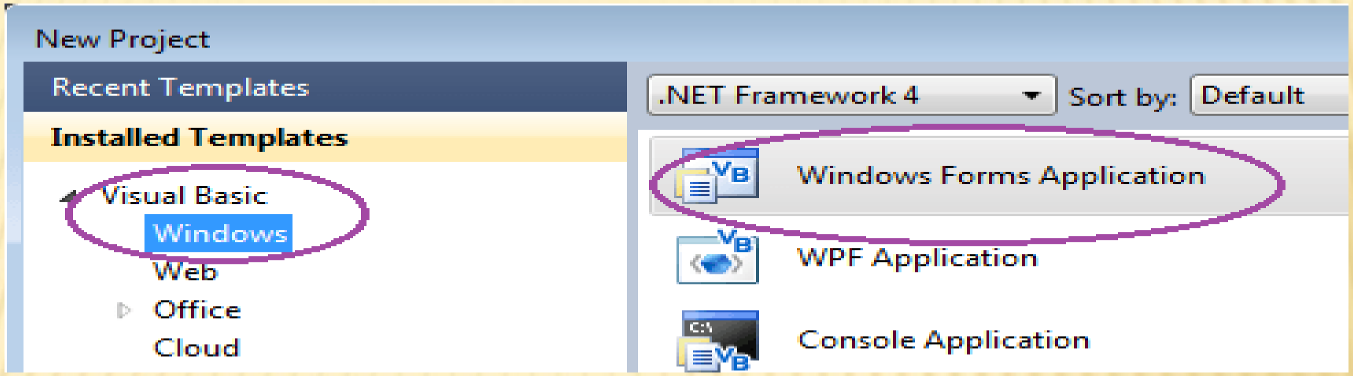
VB.Net programmers have made extensive use of forms to build user interfaces. Each time you create a Windows application, Visual Studio will display a default blank form, onto which you can drag and drop controls from the Visual Studio Toolbox window.



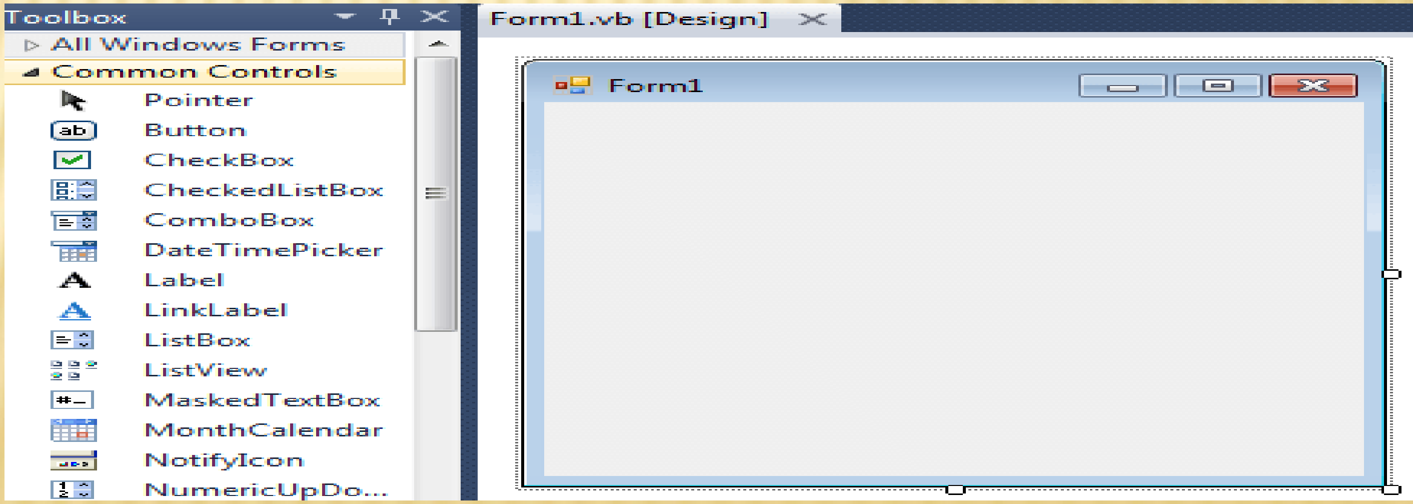
- The first step is to start a new project and build a form.
- Open your Visual Studio and select File->New Project and select Visual Basic from the New project dialog box and select Windows Forms Application.
- Enter your project name instead of WindowsApplication1 in the bottom of dialogue box and click OK button.
- The following picture shows how to crate a new Form in Visual Studio.



Select project type from New project dialog Box



When you add a Windows Form to your project, many of the forms properties are set by default. Although these values are convenient, they will not always suit your programming needs. The following picture shows how is the default Form look like.



UNIT 3:

Object Oriented Programming in vb.net

CLASS AND OBJECT

When you define a class, you define a blueprint for a data type. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

Objects are instances of a class. The methods and variables that constitute a class are called members of the class

OBJECT-ORIENTED (OO) PROGRAM

- Objects
  - Consist of one or more data values which define the state or properties of the object
  - Encapsulated by a set of functions (methods) that can be applied to that object

CONSTRUCTORS AND DESTRUCTORS

- Examine first line of code for a form in the Editor

A class **constructor** is a special member Sub of a class that is executed whenever we create new objects of that class. A constructor has the name **New** and it does not have any return type. Following program explains the concept of constructor −

```
Class Line
   Private length As Double    ' Length of a line
   Public Sub New()   'constructor
      Console.WriteLine("Object is being created")
   End Sub

   Public Sub setLength(ByVal len As Double)
      length = len
   End Sub

   Public Function getLength() As Double
      Return length
   End Function
   Shared Sub Main()
      Dim line As Line = New Line()
      'set line length
      line.setLength(6.0)
      Console.WriteLine("Length of line : {0}", line.getLength())
      Console.ReadKey()
   End Sub
End Class
```

- When the above code is compiled and executed, it produces the following result −

- Object is being created
- Length of line : 6

- A default constructor does not have any parameter, but if you need, a constructor can have parameters. Such constructors are called parameterized constructors. This technique helps you to assign initial value to an object at the time of its creation as shown in the following example −

```
Class Line
   Private length As Double    ' Length of a line
   Public Sub New(ByVal len As Double)   'parameterised constructor
      Console.WriteLine("Object is being created, length = {0}", len)
      length = len
   End Sub
   Public Sub setLength(ByVal len As Double)
      length = len
   End Sub

   Public Function getLength() As Double
      Return length
   End Function
   Shared Sub Main()
      Dim line As Line = New Line(10.0)
      Console.WriteLine("Length of line set by constructor : {0}", line.getLength())
      'set line length
      line.setLength(6.0)
      Console.WriteLine("Length of line set by setLength : {0}", line.getLength())
      Console.ReadKey()
   End Sub
End Class
```

- When the above code is compiled and executed, it produces the following result –

```
Object is being created, length = 10
Length of line set by constructor : 10
Length of line set by setLength : 6
```

DESTRUCTOR

- A destructor is a special member Sub of a class that is executed whenever an object of its class goes out of scope.
- A destructor has the name Finalize and it can neither return a value nor can it take any parameters. Destructor can be very useful for releasing resources before coming out of the program like closing files, releasing memories, etc.
- Destructors cannot be inherited or overloaded.
- Following example explains the concept of destructor −

```
Live Demo
Class Line
   Private length As Double    ' Length of a line
   Public Sub New()   'parameterised constructor
      Console.WriteLine("Object is being created")
   End Sub

   Protected Overrides Sub Finalize()  ' destructor
      Console.WriteLine("Object is being deleted")
   End Sub

   Public Sub setLength(ByVal len As Double)
      length = len
   End Sub

   Public Function getLength() As Double
      Return length
   End Function

   Shared Sub Main()
      Dim line As Line = New Line()
      'set line length
      line.setLength(6.0)
      Console.WriteLine("Length of line : {0}", line.getLength())
      Console.ReadKey()
   End Sub
End Class
```

- When the above code is compiled and executed, it produces the following result –

```
Object is being created
Length of line : 6
Object is being deleted
```

METHOD OVERLOADING

- visual basic, **Method Overloading** means defining multiple methods with the same name but with different parameters. By using **Method Overloading**, we can perform different tasks with the same method name by passing different parameters.
-
- Suppose, if we want to overload a method in visual basic, we need to define another method with the same name but with different signatures. In visual basic, the Method Overloading is also called as **compile time polymorphism** or **early binding**.
-
- Following is the code snippet of implementing a method overloading in a visual basic progra

- An exception is a problem that arises during the execution of a program. An exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

- Exceptions provide a way to transfer control from one part of a program to another. VB.Net exception handling is built upon four keywords - Try, Catch, Finally and Throw.

- Try – A Try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more Catch blocks.
- Catch – A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The Catch keyword indicates the catching of an exception.
- Finally – The Finally block is used to execute a given set of statements, whether an exception is thrown or not thrown. For example, if you open a file, it must be closed whether an exception is raised or not.
- Throw – A program throws an exception when a problem shows up. This is done using a Throw keyword.

- Syntax
- Assuming a block will raise an exception, a method catches an exception using a combination of the Try and Catch keywords. A Try/Catch block is placed around the code that might generate an exception. Code within a Try/Catch block is referred to as protected code, and the syntax for using Try/Catch looks like the following –

```
Try
    [ tryStatements ]
    [ Exit Try ]
[ Catch [ exception [ As type ] ] [ When expression ]
    [ catchStatements ]
    [ Exit Try ] ]
[ Catch ... ]
[ Finally
    [ finallyStatements ] ]
End Try
```

EXCEPTION CLASSES IN .NET FRAMEWORK

- In the .Net Framework, exceptions are represented by classes. The exception classes in .Net Framework are mainly directly or indirectly derived from the System.Exception class. Some of the exception classes derived from the System.Exception class are the System.ApplicationException and System.SystemException classes.

- The System.ApplicationException class supports exceptions generated by application programs. So the exceptions defined by the programmers should derive from this class.

- The System.SystemException class is the base class for all predefined system exception.

- The following table provides some of the predefined exception classes derived from the Sytem.SystemException class −

Exception Class	Description
System.IO.IOException	Handles I/O errors.
System.IndexOutOfRangeException	Handles errors generated when a method refers to an array index out of range.
System.ArrayTypeMismatchException	Handles errors generated when type is mismatched with the array type.
System.NullReferenceException	Handles errors generated from deferencing a null object.
System.DivideByZeroException	Handles errors generated from dividing a dividend with zero.
System.InvalidCastException	Handles errors generated during typecasting.
System.OutOfMemoryException	Handles errors generated from insufficient free memory.
System.StackOverflowException	Handles errors generated from stack overflow.

HANDLING EXCEPTIONS

- VB.Net provides a structured solution to the exception handling problems in the form of try and catch blocks. Using these blocks the core program statements are separated from the error-handling statements.

- These error handling blocks are implemented using the Try, Catch and Finally keywords. Following is an example of throwing an exception when dividing by zero condition occurs −

- Live Demo

```
Module exceptionProg
   Sub division(ByVal num1 As Integer, ByVal num2 As Integer)
      Dim result As Integer
      Try
         result = num1 \ num2
      Catch e As DivideByZeroException
         Console.WriteLine("Exception caught: {0}", e)
      Finally
         Console.WriteLine("Result: {0}", result)
      End Try
   End Sub
   Sub Main()
      division(25, 0)
      Console.ReadKey()
  End Sub
End Module
```

- Exception caught: System.DivideByZeroException: Attempted to divide by zero.
- at ...
- Result: 0

- Creating User-Defined Exceptions
- You can also define your own exception. User-defined exception classes are derived from the ApplicationException class. The following example demonstrates this −

- Live Demo

```
Module exceptionProg
  Public Class TempIsZeroException : Inherits ApplicationException
    Public Sub New(ByVal message As String)
      MyBase.New(message)
    End Sub
  End Class
  Public Class Temperature
    Dim temperature As Integer = 0
    Sub showTemp()
      If (temperature = 0) Then
        Throw (New TempIsZeroException("Zero Temperature found"))
      Else
        Console.WriteLine("Temperature: {0}", temperature)
      End If
    End Sub
  End Class
  Sub Main()
    Dim temp As Temperature = New Temperature()
    Try
      temp.showTemp()
    Catch e As TempIsZeroException
      Console.WriteLine("TempIsZeroException: {0}", e.Message)
    End Try
    Console.ReadKey()
  End Sub
End Module
```

- When the above code is compiled and executed, it produces the following result −
- TempIsZeroException: Zero Temperature found

- Throwing Objects
- You can throw an object if it is either directly or indirectly derived from the System.Exception class.

- You can use a throw statement in the catch block to throw the present object as −

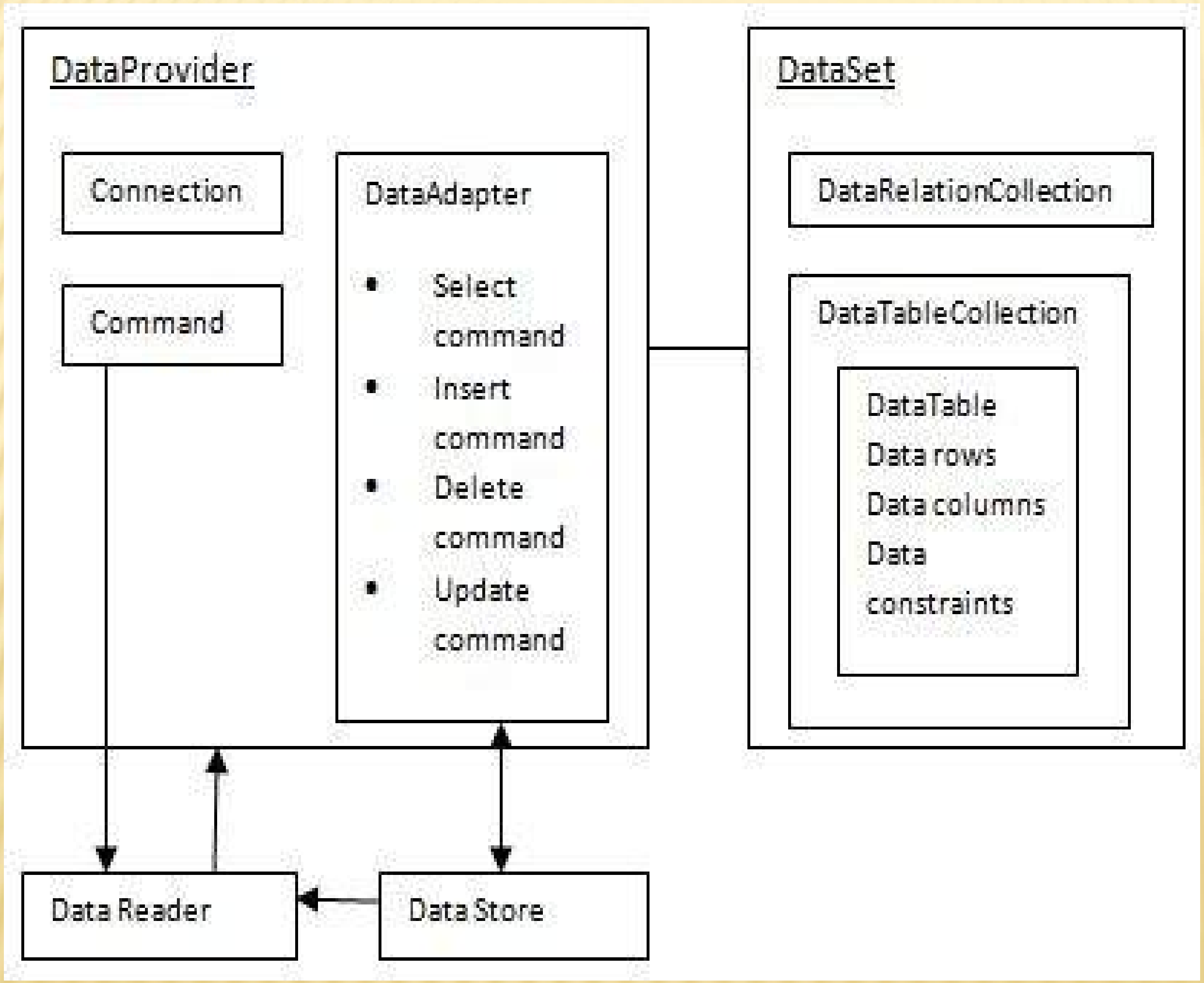
- Throw [expression]
- The following program demonstrates this −

```
Module exceptionProg
   Sub Main()
      Try
         Throw New ApplicationException("A custom exception _ is being thrown here...")
      Catch e As Exception
         Console.WriteLine(e.Message)
      Finally
         Console.WriteLine("Now inside the Finally Block")
      End Try
      Console.ReadKey()
   End Sub
End Module
```

UNIT :5

- Architecture Of ADO.Net

- ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.


DataProvider
Connection
Command
DataAdapter
• Select command
• Insert command
• Delete command
• Update command
DataSet
DataRelationCollection
DataTableCollection
DataTable
Data rows
Data columns
Data constraints
Data Reader
Data Store

THE DATASET CLASS

- The DataSet Class
- The dataset represents a subset of the database. It does not have a continuous connection to the database. To update the database a reconnection is required. The DataSet contains DataTable objects and DataRelation objects. The DataRelation objects represent the relationship between two tables.

- Following table shows some important properties of the DataSet class:

Properties	Description
CaseSensitive	Indicates whether string comparisons within the data tables are case-sensitive.
Container	Gets the container for the component.
DataSetName	Gets or sets the name of the current data set.
DefaultViewManager	Returns a view of data in the data set.
DesignMode	Indicates whether the component is currently in design mode.
EnforceConstraints	Indicates whether constraint rules are followed when attempting any update operation.
Events	Gets the list of event handlers that are attached to this component.
ExtendedProperties	Gets the collection of customized user information associated with the DataSet.
HasErrors	Indicates if there are any errors.
IsInitialized	Indicates whether the DataSet is initialized.
Locale	Gets or sets the locale information used to compare strings within the table.
Namespace	Gets or sets the namespace of the DataSet.
Prefix	Gets or sets an XML prefix that aliases the namespace of the DataSet.
Relations	Returns the collection of DataRelation objects.
Tables	Returns the collection of DataTable objects.

THE DATATABLE CLASS

The DataTable class represents the tables in the database. It has the following important properties; most of these properties are read only properties except the PrimaryKey property:
The following table shows some important methods of the DataTable class:

Properties	Description
ChildRelations	Returns the collection of child relationship.
Columns	Returns the Columns collection.
Constraints	Returns the Constraints collection.
DataSet	Returns the parent DataSet.
DefaultView	Returns a view of the table.
ParentRelations	Returns the ParentRelations collection.
PrimaryKey	Gets or sets an array of columns as the primary key for the table.
Rows	Returns the Rows collection.

Methods	Description
AcceptChanges	Commits all changes since the last AcceptChanges.
Clear	Clears all data from the table.
GetChanges	Returns a copy of the DataTable with all changes made since the AcceptChanges method was called.
GetErrors	Returns an array of rows with errors.
ImportRows	Copies a new row into the table.
LoadDataRow	Finds and updates a specific row, or creates a new one, if not found any.
Merge	Merges the table with another DataTable.
NewRow	Creates a new DataRow.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.
Reset	Resets the table to its original state.
Select	Returns an array of DataRow objects.

THE DATAROW CLASS

- The DataRow object represents a row in a table. It has the following important properties:

Properties	Description
HasErrors	Indicates if there are any errors.
Items	Gets or sets the data stored in a specific column.
ItemArrays	Gets or sets all the values for the row.
Table	Returns the parent table.

The following table shows some important methods of the DataRow class:

Methods	Description
AcceptChanges	Accepts all changes made since this method was called.
BeginEdit	Begins edit operation.
CancelEdit	Cancels edit operation.
Delete	Deletes the DataRow.
EndEdit	Ends the edit operation.
GetChildRows	Gets the child rows of this row.
GetParentRow	Gets the parent row.
GetParentRows	Gets parent rows of DataRow object.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.

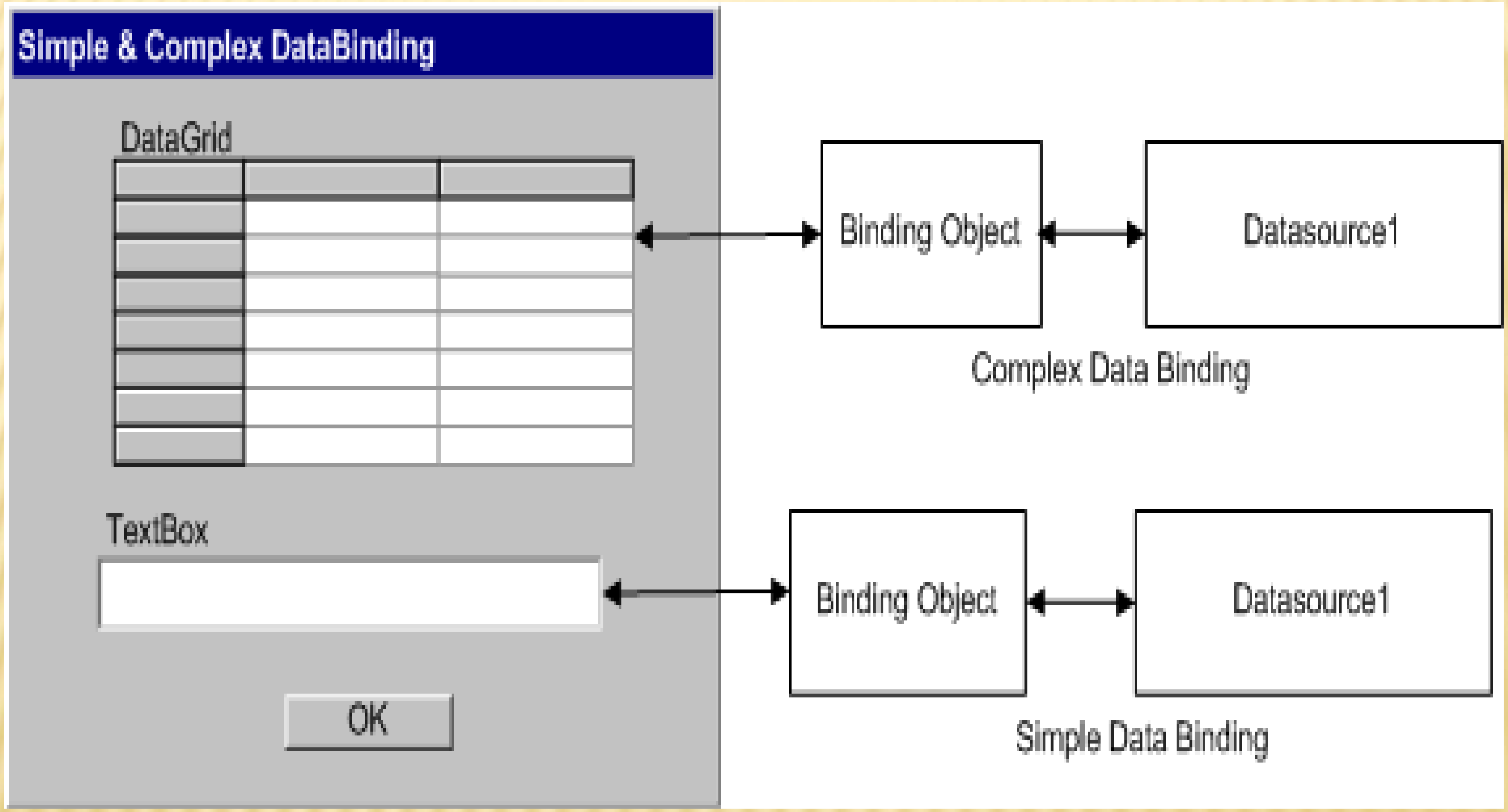
- **The DataAdapter Object**
- The DataAdapter object acts as a mediator between the DataSet object and the database. This helps the Dataset to contain data from multiple databases or other data source.
- **The DataReader Object**
- The DataReader object is an alternative to the DataSet and DataAdapter combination. This object provides a connection oriented access to the data records in the database. These objects are suitable for read-only access, such as populating a list and then breaking the connection.

DBCOMMAND AND DBCONNECTION OBJECTS

- The DbConnection object represents a connection to the data source. The connection could be shared among different command objects.

- The DbCommand object represents the command or a stored procedure sent to the database from retrieving or manipulating data.

- What is DataBinding?
- DataBinding is a powerful feature provided by the .NET Framework that enables visual elements in a client to connect to a datasource such as DataSets, DataViews, Arrays, etc. Some of the visual elements in the client can be TextBox, Datagrid, etc. A two-way connection is established such that any changes made to the datasource are reflected immediately in the visual element and vice versa.
- Below is a graphical description of the concept of databinding:
-


Simple & Complex DataBinding
DataGrid
TextBox
OK
Binding Object
Datasource1
Complex Data Binding
Binding Object
Datasource1
Simple Data Binding

- DataBinding before .NET
- In the earlier databinding models, the datasource that could be used was usually limited to a database. All DBMS systems provided their own APIs to help in building GUI applications and quickly bind them to the data. Programmer did not have the flexibility to control the databinding process with the result that most developers avoided the use of databinding.

- DataBinding with .NET
- The .NET Framework provides a very flexible and powerful approach to databinding and allows the programmer to have a fine control over the steps involved in the whole process. One of the biggest improvements with .NET has been the introduction of databinding to web pages through the use of .NET server-side web controls. Hence, building data driven web applications has been greatly simplified. Please note that this article only deals with data binding in .NET Windows Forms.

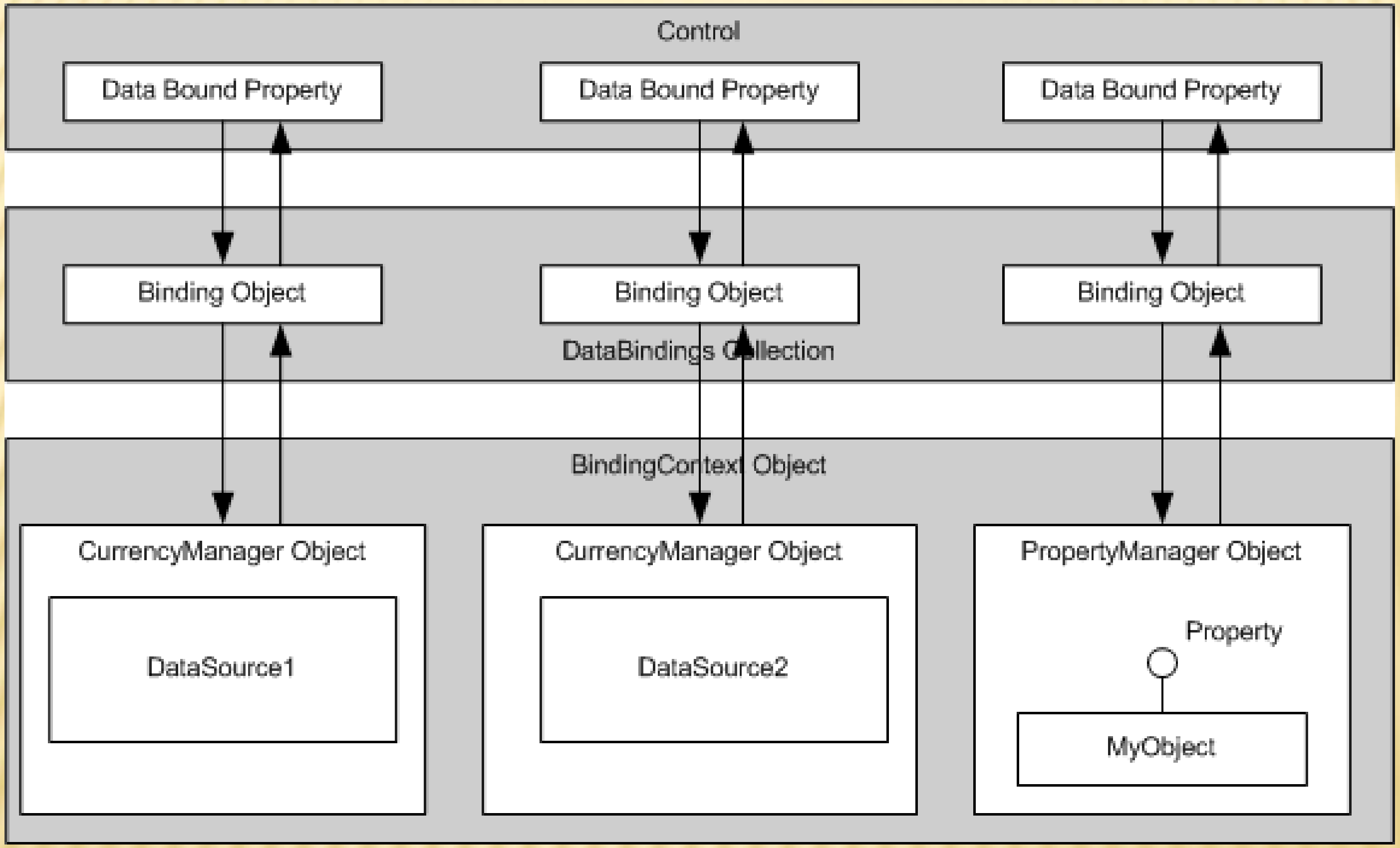
ADVANTAGES OF DATABINDING

- Databinding in .NET can be used to write data driven applications quickly. .NET data binding allows you to write less code with fast execution but still get the work done in the best way.
- .NET automatically writes a lot of databinding code for you in the background (you can see it in "Windows Generated Code" section), so the developer does not have to spend time writing code for basic databinding, but still has the flexibility of modifying any code that he would like to. We get the benefits of bound as well as unbound approach.
- Control over the Databinding process by using events. This is discussed in more detail later in the article.

DISADVANTAGES OF DATABINDING

- More optimized code can be written by using the unbound or traditional methods.
- Complete flexibility can only be achieved by using the unbound approach.

DATAFLOW DURING DATABINDING



Unit 5 : Crystal Report

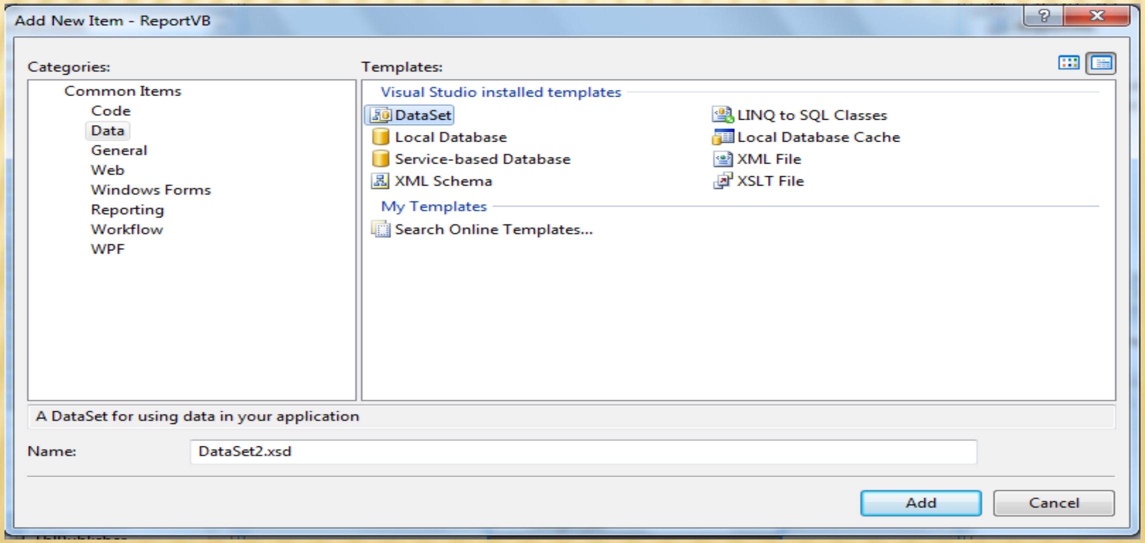
CRYSTAL REPORT

- Crystal Report that enables you to generate reports to show your data retrieved from a database
- how to get data stored in a table of a Microsoft SQL SERVER database to display on a form of your project.

- Suppose that you want to create a report to display the data from TblProduct of SaleAndStock database. This database exists in Microsoft SQL SERVER(in my machine, i have SQL SEVER 2005). The user that has a right to login to the database is called sa and its password is 123. This login information is useful when we connect the database using VB.NET code. The steps below will help you to get thing done.

STEP 1: ADD A DATASET TO THE PROJECT

Now create a new project in Visual Studio 2008. In my case, i created a project called ReportVB. Right-click the project name->Add->New Item. Under Categories, select Data and under Templates select Dataset.


Add New Item - ReportVB

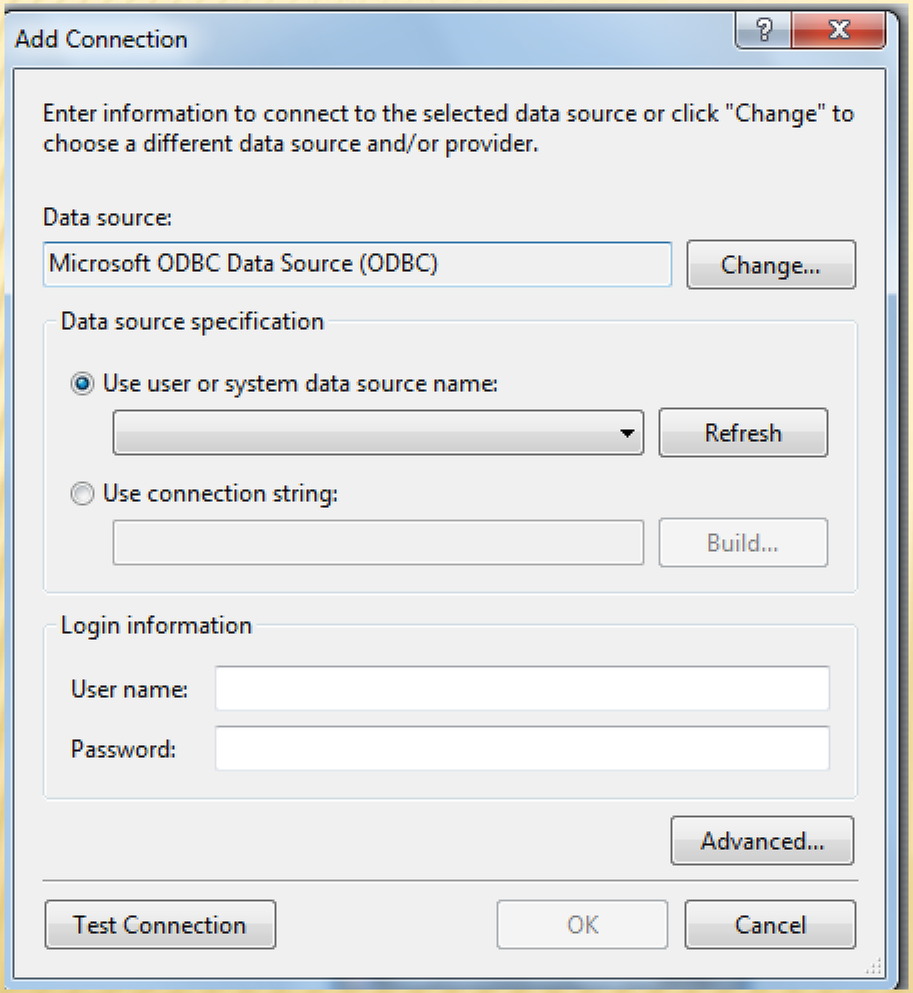
Categories:
Common Items
Code
Data
General
Web
Windows Forms
Reporting
Workflow
WPF

Templates:
Visual Studio installed templates
DataSet
Local Database
Service-based Database
XML Schema
LINQ to SQL Classes
Local Database Cache
XML File
XSLT File
My Templates
Search Online Templates...

A DataSet for using data in your application

Name: DataSet2.xsd

- On the Server Explorer next to the active dataset window, click Connect to Database icon to add a connection to the database server. On the Add Connection dialog, click Change button of the Data source block to show the Change Data Source dialog



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft ODBC Data Source (ODBC)

Data source specification

Use user or system data source name: [✓]

Use connection string:

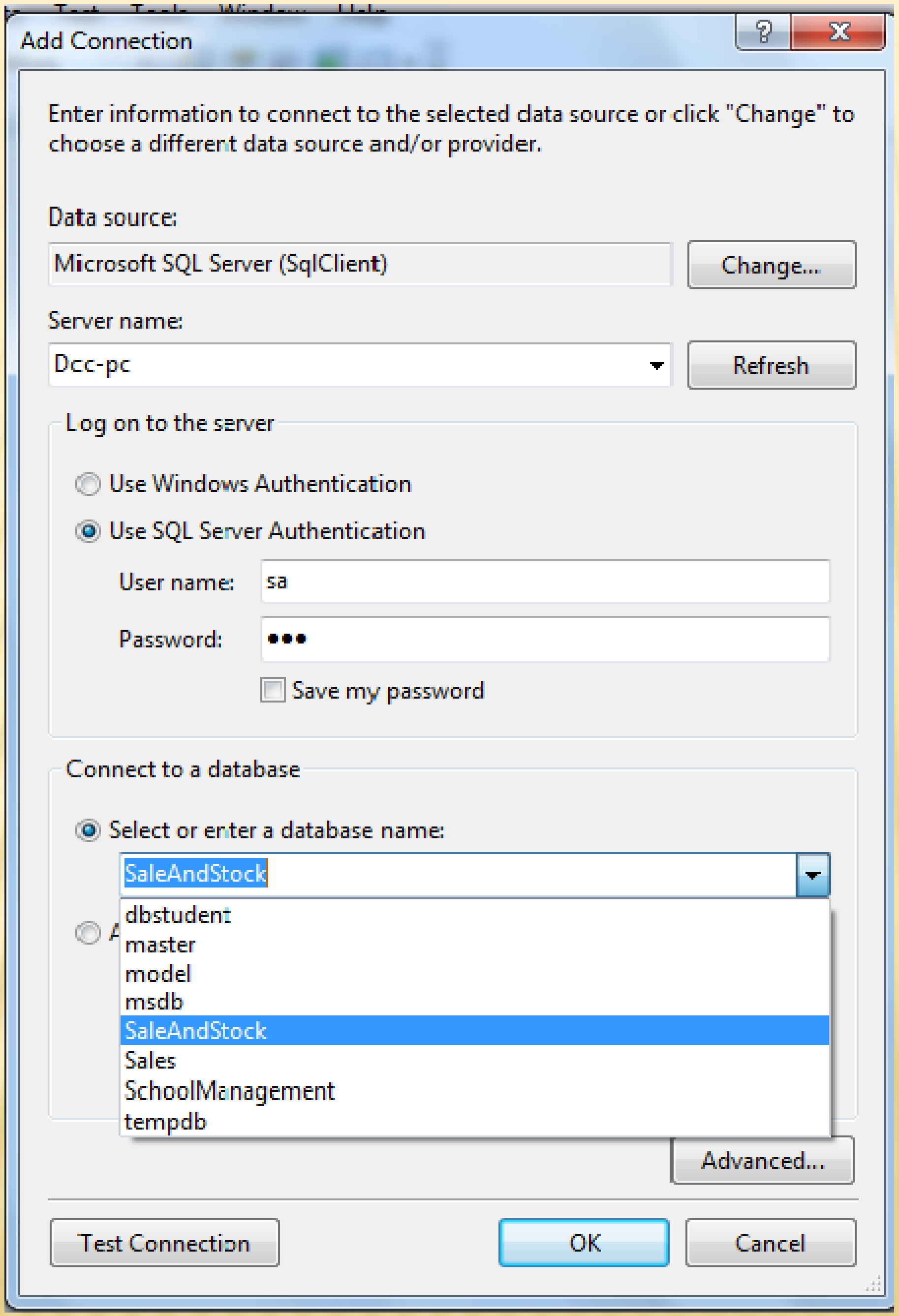
Login information

User name:

Password:

- Then on the Change Data Source dialog, select Microsoft SQL Server. After that you will see a new Add Connection dialog that shows the Server name box. In this box, you need to enter the name of your SQL SERVER. In my case, this name is Dcc-pc.
- +In the Log on to the server block, select Use SQL Server Authentication. Then type the user name and password in to the User name box and Password box.

- In the Connection to a database, select Select or enter a database name option. Then click the dropdown list to select SaleAndStock database(a sample database used in this tutorial).



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient)

Server name:
Dcc-pc

Log on to the server

( ) Use Windows Authentication
(●) Use SQL Server Authentication

User name: sa
Password: •••
[ ] Save my password

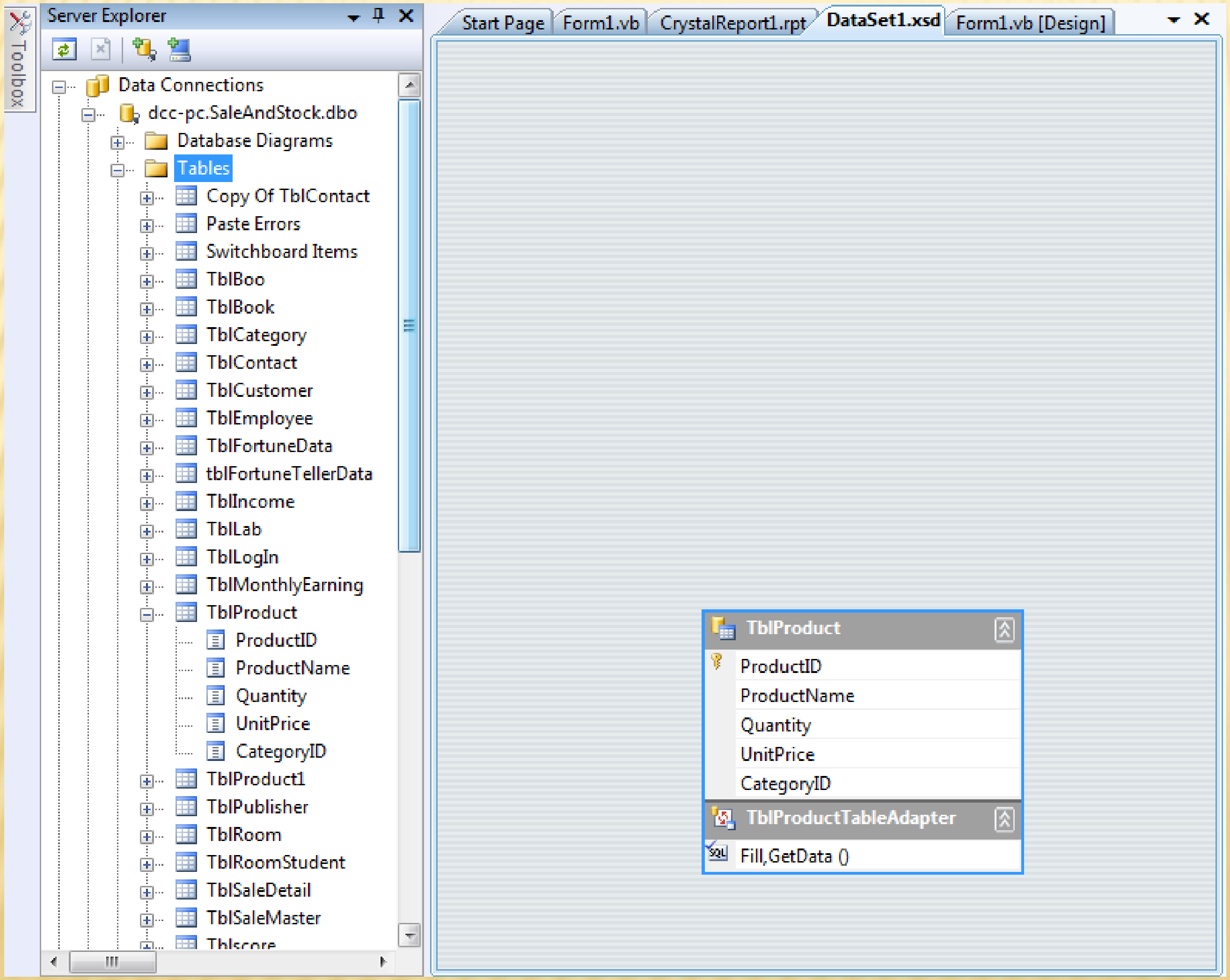
Connect to a database

(●) Select or enter a database name:
SaleAndStock

- dbstudent
- master
- model
- msdb
- SaleAndStock
- Sales
- SchoolManagement
- tempdb

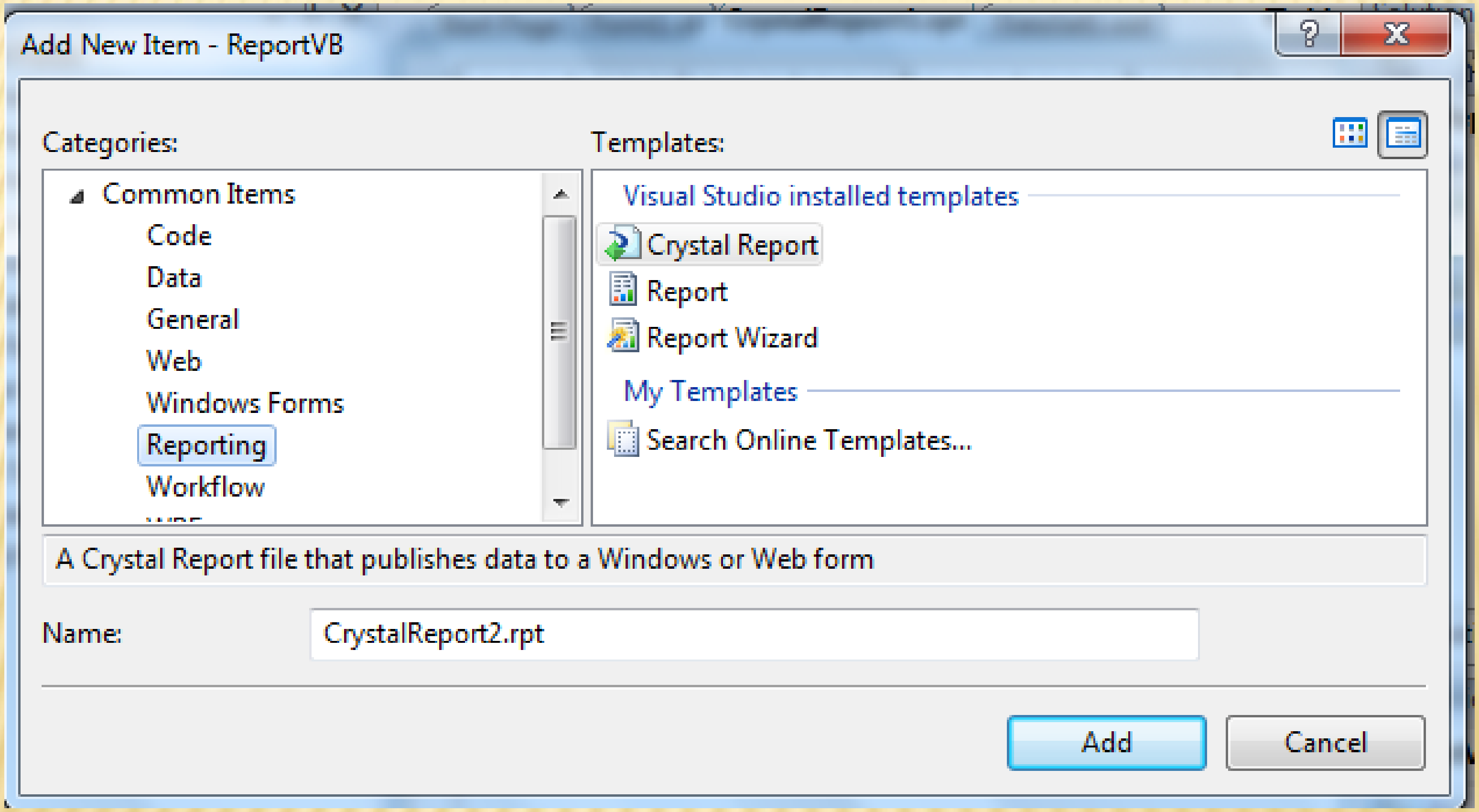
- Click Test Connection button to test the connection to the SQL SERVER. If there is a dialog showing that the Connection is succeed, it means that you can move to the next step. If an error occurs you need to check what you have provided again.

- Step 2: Add the TblProduct table to the dataset
- After you connected successfully to the SQL Server database, look Server Explorer again and click Data Connection to expand it then select the connection that you have created in the previous step. Then click Tables to expand and drag the TblProduct table to the dataset designer window.


Toolbox
Server Explorer
Data Connections
dcc-pc.SaleAndStock.dbo
Database Diagrams
Tables
Copy Of TblContact
Paste Errors
Switchboard Items
TblBoo
TblBook
TblCategory
TblContact
TblCustomer
TblEmployee
TblFortuneData
tblFortuneTellerData
TblIncome
TblLab
TblLogIn
TblMonthlyEarning
TblProduct
ProductID
ProductName
Quantity
UnitPrice
CategoryID
TblProduct1
TblPublisher
TblRoom
TblRoomStudent
TblSaleDetail
TblSaleMaster
Tblscore
Start Page
Form1.vb
CrystalReport1.rpt
DataSet1.xsd
Form1.vb [Design]
TblProduct
ProductID
ProductName
Quantity
UnitPrice
CategoryID
TblProductTableAdapter
Fill,GetData ()

- Step 3: Add a Crystal Report and link to the table of dataset
- Now you have a schema of a dataset that connects to the TblProduct of the SaleAndStock database located in SQL Server. The next step is to add a Crystal Report item in to your project.

- Right-click the project name(ReportVB)->Add->New Item. On the New Item dialog, under Categories select Reporting and under Templates select. You will see the Crystal Reports Gallery dialog. Select As a Blank Report to create a blank report and accept the default name(CrystalReport1).


Add New Item - ReportVB

Categories:

- Common Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms
  - Reporting
  - Workflow

Templates:

Visual Studio installed templates

- Crystal Report
- Report
- Report Wizard

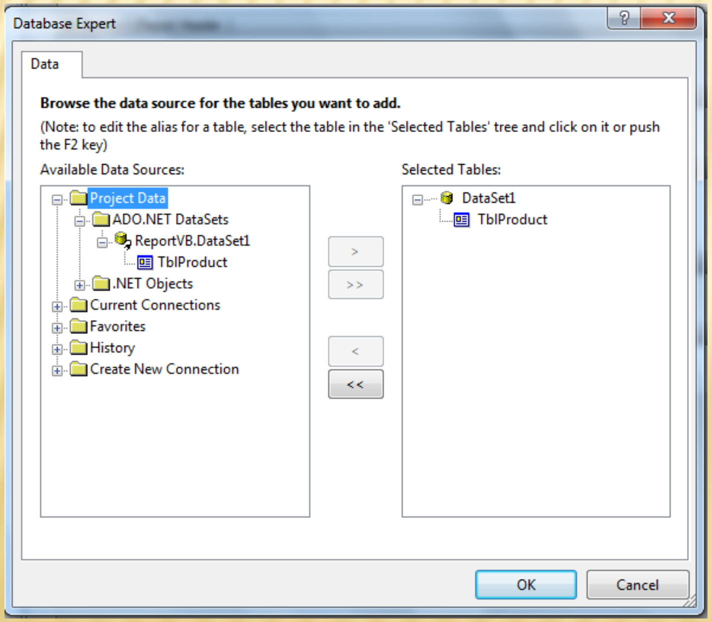
My Templates

- Search Online Templates...

A Crystal Report file that publishes data to a Windows or Web form

Name: CrystalReport2.rpt

- To link to the table of the dataset, right-click Field Explorer next to the active window of CrystalReport1 then select Database Expert. You will see the dialog as shown below:



Browse the data source for the tables you want to add.

(Note: to edit the alias for a table, select the table in the 'Selected Tables' tree and click on it or push the F2 key)

Available Data Sources:

```
Project Data
  ADO.NET DataSets
    ReportVB.DataSet1
      TblProduct
  .NET Objects
Current Connections
Favorites
History
Create New Connection
```

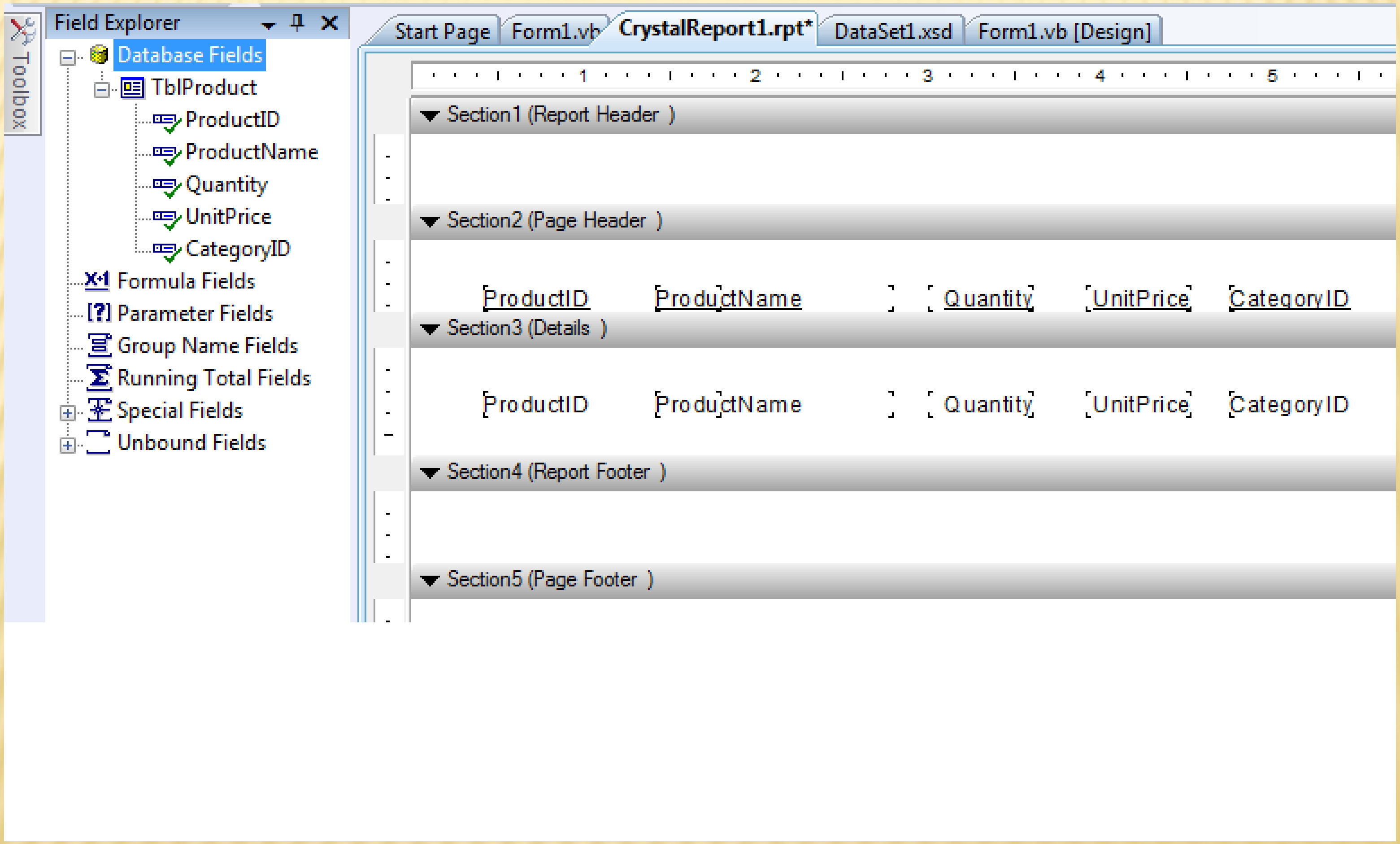
Selected Tables:

```
DataSet1
  TblProduct
```

CRYST

- Select Project Data and select ADO.NET DataSets, then you will see the TblProduct table. Select this table and click the arrow button to add the table to the Selected Table area. Click OK.

- +Now back to the active window of the CrystalReport1 and drag fields of TblProduct table that you want to the designer area of the CrystalReport1.


Field Explorer

- Database Fields
  - TblProduct
    - ProductID
    - ProductName
    - Quantity
    - UnitPrice
    - CategoryID
- Formula Fields
- Parameter Fields
- Group Name Fields
- Running Total Fields
- Special Fields
- Unbound Fields

Start Page | Form1.vb | CrystalReport1.rpt* | DataSet1.xsd | Form1.vb [Design]

Section1 (Report Header)

Section2 (Page Header)

ProductID ProductName Quantity UnitPrice CategoryID

Section3 (Details)

ProductID ProductName Quantity UnitPrice CategoryID

Section4 (Report Footer)

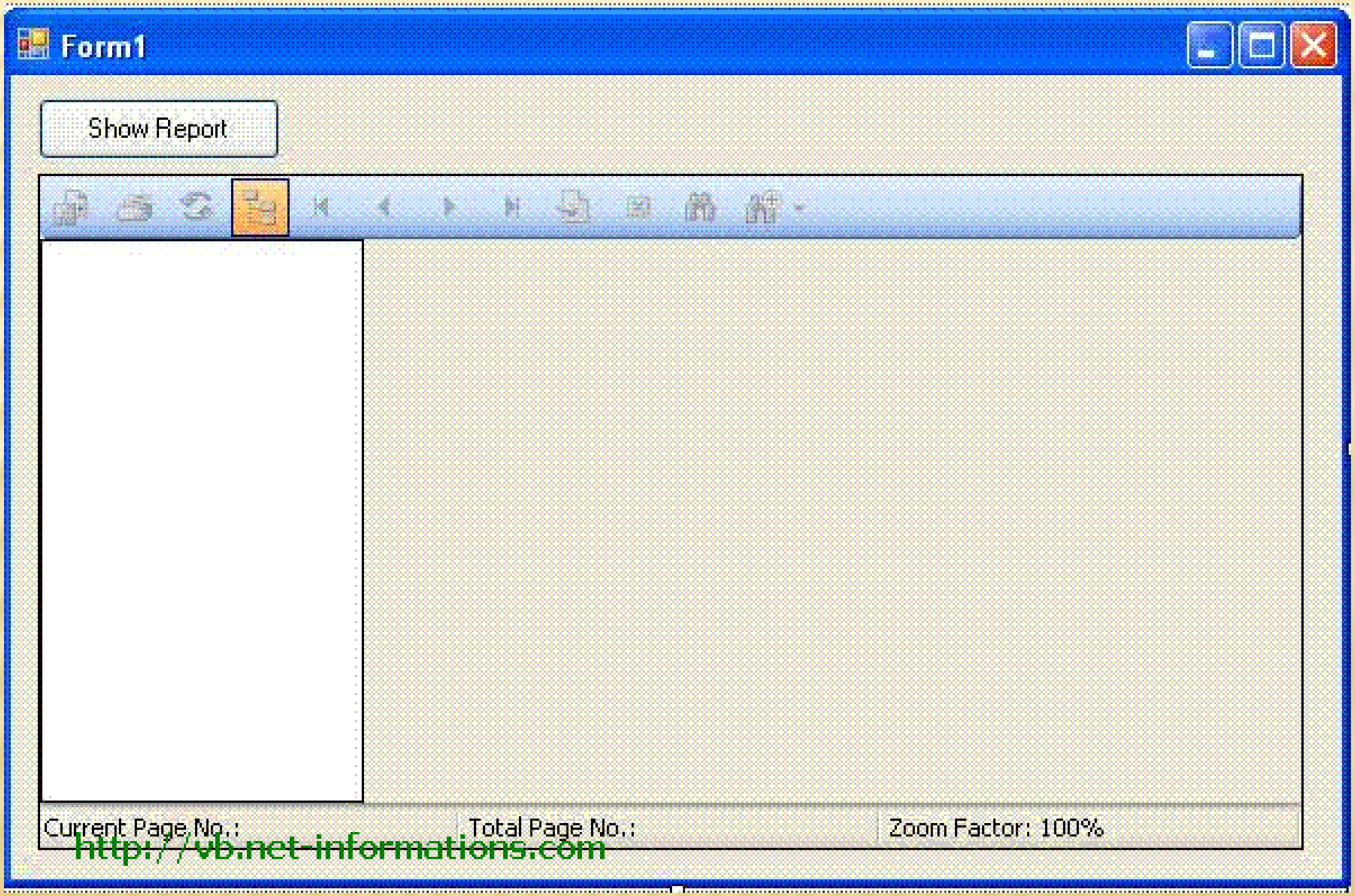
Section5 (Page Footer)

- Step 4: Add a Crystal Report Viewer control to the form of the project(look for this control in the Toolbox) and writing VB.NET code on the form load event.

- +Accept the default name of the Crystal Report Viewer(CrystalReportVewer1)

- Open the code window by double-click the form and in the Form1_load procedure write the following code:
- Dim rpt As New CrystalReport1() 'The report you created.
- Dim myCon As SqlConnection
- Dim myAdapter As SqlDataAdapter
- Dim myDataset As New DataSet1() 'The DataSet you created.
- myCon = New SqlConnection("Server=(local);Database=SaleAndStock;User=sa; Password=123;")
- myAdapter = New SqlDataAdapter("Select * FROM TblProduct", myCon)
- myAdapter.Fill(myDataset, "TblProduct")
- rpt.SetDataSource(myDataset)
- CrystalReportViewer1.ReportSource = rpt
- You also need to add the line of code below to the general declaration section of form module:
- Imports System.Data.SqlClient
- Therefore, the full code becomes as shown below:
- Imports System.Data.SqlClient
- Public Class Form1
- Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
- Dim rpt As New CrystalReport1() 'The report you created.
- Dim myCon As SqlConnection
- Dim myAdapter As SqlDataAdapter
- Dim myDataset As New DataSet1() 'The DataSet you created.
- myCon = New SqlConnection("Server=(local);Database=SaleAndStock;User=sa; Password=123;")
- myAdapter = New SqlDataAdapter("Select * FROM TblProduct", myCon)
- myAdapter.Fill(myDataset, "TblProduct")
- rpt.SetDataSource(myDataset)
- CrystalReportViewer1.ReportSource = rpt
- End Sub
- End Class
- +Run your project and see the result.

- Now the designing part is over and the next step is to call the created Crystal Reports in VB.NET through Crystal Reports Viewer control .
- Select the default form (Form1.vb) you created in VB.NET and drag a button and CrystalReportViewer control to your form.



Current Page No.: Total Page No.: Zoom Factor: 100%

http://vb.net-informations.com

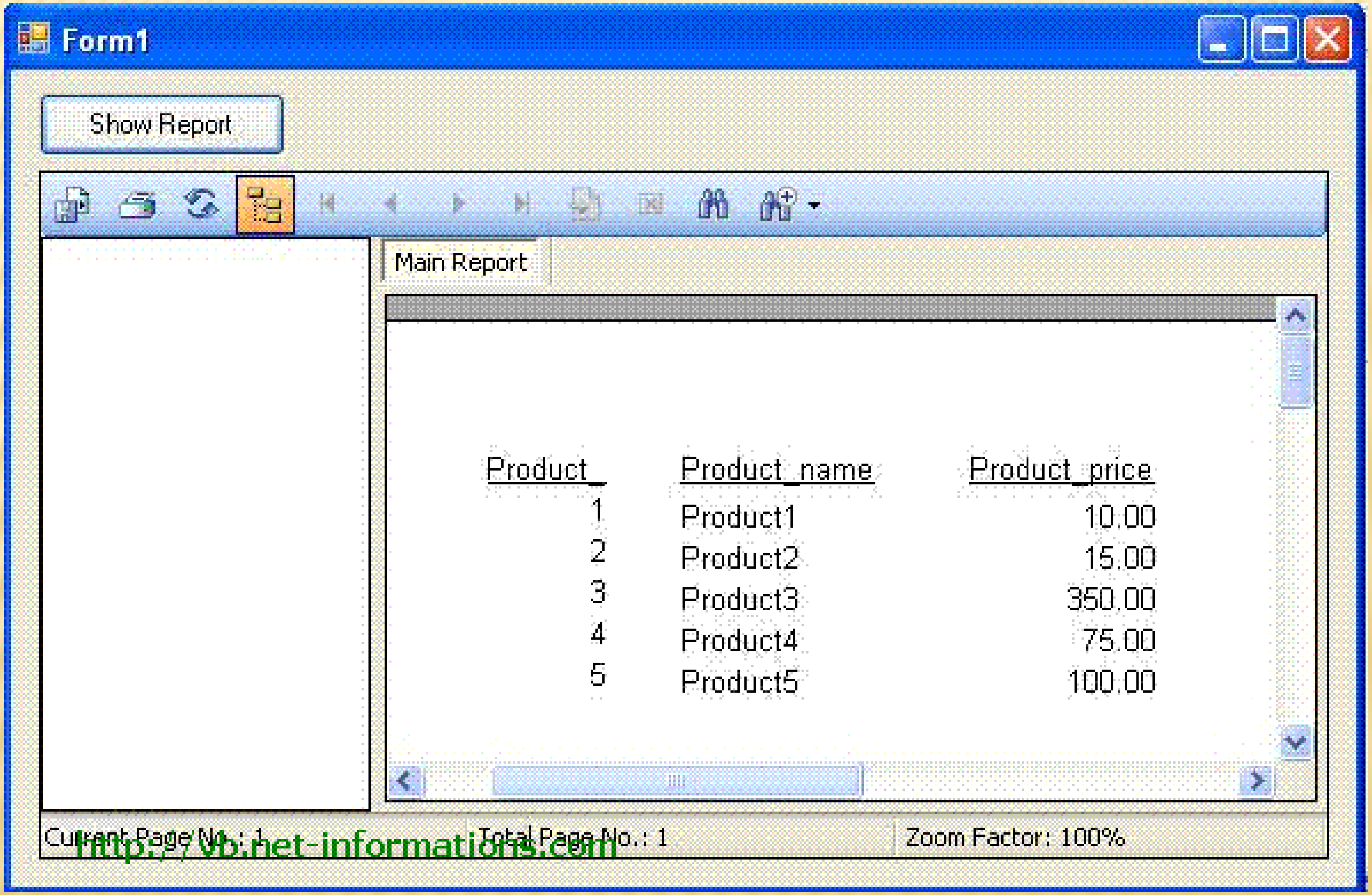
- Select Form's source code view and put the code on top.
-
- Imports CrystalDecisions.CrystalReports.Engine
- Put the following source code in the button click event

```
Imports CrystalDecisions.CrystalReports.Engine
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object,
        ByVal e As System.EventArgs) Handles Button1.Click
         Dim cryRpt As New ReportDocument
         cryRpt.Load("PUT CRYSTAL REPORT PATH
HERE\CrystalReport1.rpt")
         CrystalReportViewer1.ReportSource = cryRpt
         CrystalReportViewer1.Refresh()
    End Sub
End Class
```

- **NOTES:**
  cryRpt.Load("PUT CRYSTAL REPORT PATH HERE\CrystalReport1.rpt")

  The Crystal Reports is in your project location, there you can see **CrystalReport1.rpt** . So give the full path name of report here.

  After you run the source code you will get the report like this.



Product_	Product_name	Product_price
1	Product1	10.00
2	Product2	15.00
3	Product3	350.00
4	Product4	75.00
5	Product5	100.00

Current Page No.: 1 Total Page No.: 1 Zoom Factor: 100%

http://vb.net-informations.com

DACC

DNYANSAGAR ARTS AND COMMERCE COLLEGE

DACC
DNYANSAGAR
ARTS AND COMMERCE
COLLEGE

DACC

DNYANSAGAR
ARTS AND COMMERCE
COLLEGE