Web Service

V.Mareeswari
Assistant Professor (Sr)
SITE, VIT University
Cabin No: SJT210-A30

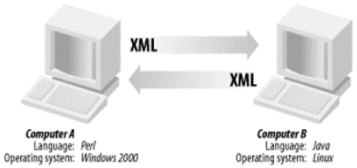
Web Applications vs Web Services

Web Application
- User-to-program interaction
- Static integration of components
- Monolithic service
- Ad hoc or proprietary protocol

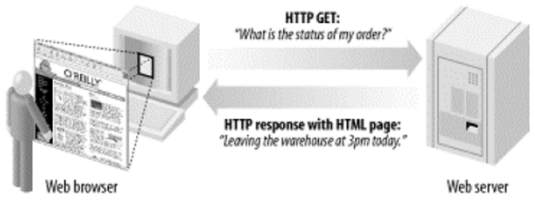
Web Services
- Program-to-program interaction
- Dynamic integration of components
- Service aggregation
- Interoperability

Web Service

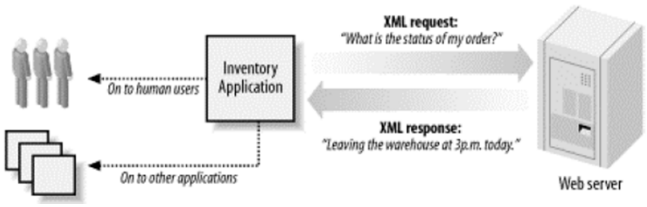
- main implementation technology for SOA.
- a *wrapper technology* - It only extends them with interoperability among heterogeneous platforms(OS, Programming Language).
- Whether a Web service performs well heavily depends on which server technology is used to implement its business logics.



Human-centric Web

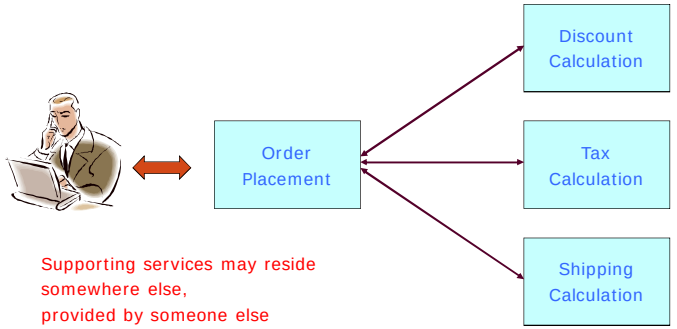


Application-centric Web



5 V.Mareeswari / AP(Sr) / SITE / VITU 10-08-2016

Order Placement



Supporting services may reside somewhere else, provided by someone else

6 V.Mareeswari / AP(Sr) / SITE / VITU 10-08-2016

Web Service

- The term *Web services* describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone.
- XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available.

7 V.Mareeswari / AP(Sr) / SITE / VITU 10-08-2016

Web Services Architecture

8 V.Mareeswari / AP(Sr) / SITE / VITU 10-08-2016

.NET

- .NET was originally called NGWS (Next Generation Windows Services).
- .NET is Microsoft's new Internet and Web based infrastructure
- .NET is NOT a new operating system
- .NET delivers software as Web Services
- .NET is a framework for universal services
- .NET is a server centric computing model
- .NET will run in any browser on any platform

.NET Framework

- The .NET Framework is a common environment for building, deploying, and running Web Services and Web Applications.
- It is language neutral. Currently it supports C++, C#, Visual Basic, JScript (The Microsoft version of JavaScript) and COBOL. Third-party languages - like Eiffel, Perl, Python, Smalltalk, and others - will also be available for building future .NET Framework applications.
- It provides a feature-rich application execution environment, simplified development and easy integration between a number of different development languages.

Type of Applications can be developed by .NET are

1. **ASP.NET Web applications:** These include dynamic and data driven browser based applications.
2. **Windows Form based applications:** These refer to traditional rich client applications.
3. **Console applications:** These refer to traditional DOS kind of applications like batch scripts.
4. **Component Libraries:** This refers to components that typically encapsulate some business logic.
5. **Windows Custom Controls**: As with traditional ActiveX controls, you can develop your own windows controls.
6. **Web Custom Controls:** The concept of custom controls can be extended to web applications allowing code reuse and modularization.
7. **Web services:** They are "web callable" functionality available via industry standards like HTTP, XML and SOAP.
8. **Windows Services:** They refer to applications that run as services in the background. They can be configured to start automatically when the system boots

.NET Programming Language

- A Sharp (.NET)
- Ada
- APL
- Axum
- Boo
- C Sharp
- C++/CLI
- COBOL
- Cobra
- Cola
- Component Pascal
- F Sharp
- F*
- Fantom
- IronLisp
- IronPython
- IronRuby
- IronScheme
- J Sharp
- JavApi
- JScript .NET
- L Sharp
- Managed Extensions for C++
- Mercury
- Microsoft Small Basic
- Microsoft Visual C Sharp
- Nemerle
- Oxygene P Sharp
- Script.NET
- VistaSmalltalk
- Visual Basic .NET
- Windows PowerShell
- Zonnon

Introduction to C#

- C# is defined as a simple, modern, object-oriented, and type-safe programming language derived from C and C++.
- Developed by Anders Hejlsberg of Microsoft especially for the .NET platform, C# derives its features from a number of languages like C, C++, and Java.
- Specifically written to offer the simplicity of Visual Basic and power of C++ as an object-oriented language, C# makes it easier for developers to create, debug, and deploy enterprise applications.
- It has also been predicted that C# will become the favored language for developing applications on the .NET platform.

Introduction to Visual Studio

- The Visual Studio.NET IDE has also been given a facelift and packed with a wide variety of new functionalities.
- A bitmap editor, debugger, Web Forms designer, Windows Forms designer,Web Services designer, XML editor, HTML editor,Web browser, Server Resources Explorer, and multi-language support have all been packed into one single IDE.

Integrated Development Environment (IDE) for C#

Microsoft provides the following development tools for C# programming:

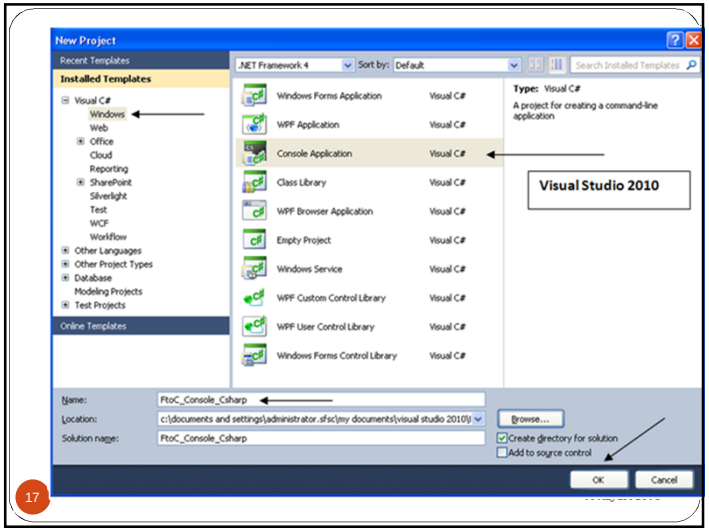
- Visual Studio 2010 (VS)
- Visual C# 2010 Express (VCE)
- Visual Web Developer

The last two are freely available from Microsoft official website.

Building C# applications in Visual Studio 2010

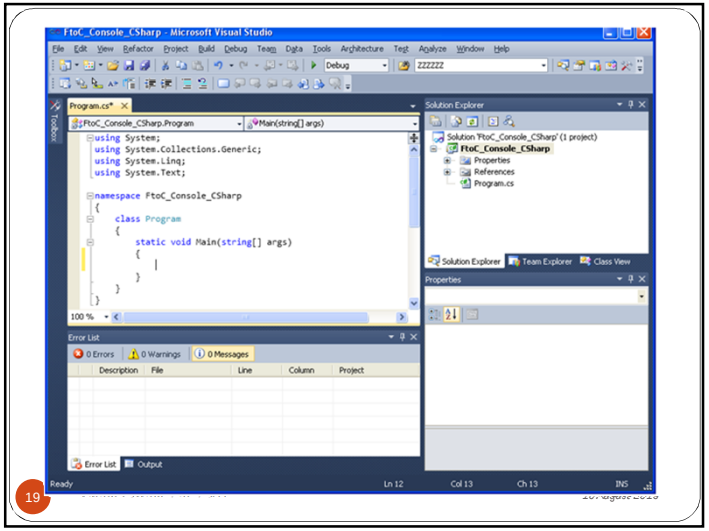
The steps required to create this project are given below:

1. With Visual Studio running, pull down the File menu and choose **New Project**
2. Highlight Visual C# on the left hand side and then select "**Windows**" on the left side
3. Select **Console Application** on the right hand side.
4. Give the new application a name: FtoC_Console_CSharp
5. (Optional) The location for this new project can be changed by clicking on the Browse button and choosing a new folder. For example, a different folder on a local hard disk or on a USB / flash drive.
6. Click the OK button when done.



7. The new project will be created, and a new file called Program.cs will be created with some sample code to get your program started.

- Solution Explorer – lists all of the files and resources that are part of this project
- Properties – shows the properties available for the object that is currently selected
- Output window – shows errors or other details when the application is compiled (Build step). If you cannot see the Output window, pull down the **View** menu and select **Output**.




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FtoC_Console_CSharp
{
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```

Namespaces that will be used for the new program by default

The namespace representing the program being written. Uses the same name as the Project

The "Main" method that will run first when the program executes

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FtoC_Console_CSharp
{
    class Program
    {
        // The main method that will run when the program starts
        static void Main(string[] args)
        {
                double tempf;     // Declare a floating point variable
                double tempc;     // Declare a floating point variable
                // Output a prompt for the user to see
                Console.WriteLine("Enter a temperature in Fahrenheit ");
                // Accept input from the user, convert it to a double data type
                // and store it in the tempf variable
                tempf = Convert.ToDouble(Console.ReadLine());
                // Do the math to convert to Celsius
                tempc = (tempf - 32.0) * 5.0 / 9.0;
                // Output the new temperature for the user to see
                Console.WriteLine("Temperature in Celsius is " + tempc);
                // Pause at the end until the user presses a key
                Console.ReadKey();
        }  // end of Main
    } // end of class Program
} // end of namespace FtoC_Console_CSharp
```

21

- Be certain to save the file once the code is added. Pull down **File** menu and choose **Save** menu item
- Pull down the **Build** menu and choose the **Build FtoC_Console_CSharp** menu item.
- To run the program, pull down the **Debug** menu and choose **Start Debugging** menu item. The Console program will open up a new window and the program will run as shown below:

22 V.MAREESWARI / AP / SITE 10 August 2016

C#

- C# is case sensitive.
- All statements and expression must end with a semicolon (;).
- The program execution starts at the Main method.
- Console.WriteLine("Size of int: {0}", sizeof(int));

Introducing Data Types:

- *Value types* are the typical primitive types available in most programming languages and are allocated on the stack.
- *Reference types* are typically class instances and are allocated on the heap.
- Value Types = Primitive Data Types and structures
- Primitive Data Types→ char(2), int(4), float(4), double(8)

23 V.Mareeswari / AP(Sr) / SITE / VITU 10-08-2016

- object obj;
- obj = 100; / / this is boxing & reverse is unboxing
- / / cast double to int.
- i = (int)d;
- Console.WriteLine(i.ToString());
- **ToChar , ToInt16 , ToInt32,ToInt64,ToDouble,ToDateTime**
- i = Convert.ToInt32(Console.ReadLine());
- String literals → "" or with @""
- **Operators:**
- Arithmetic, Relational, Logical, Bitwise, Assignment, Misc Operators

24 V.Mareeswari / AP(Sr) / SITE / VITU 10-08-2016

Operator	Description	Example
sizeof()	Returns the size of a data type.	sizeof(int), returns 4.
typeof()	Returns the type of a class.	typeof(StreamReader);
&	Returns the address of an variable.	&a; returns actual address of the variable.
*	Pointer to a variable.	*a; creates pointer named 'a' to a variable.
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y
is	Determines whether an object is of a certain type.	If(Ford is Car) / / checks if Ford is an object of the Car class.
as	Cast without raising an	Object obj = new

Control Statements:

- if, if..else, if..else if…else, switch, nested switch statements

Looping Statements:

- While, do..while, for

Loop Control Statements

- break, continue

Infinite Loop: for (; ;)

Encapsulation

- **Access specifiers:**
  Public,Private,Protected,Internal,Protected internal
- any member with internal access specifier can be accessed from any class or method defined within the application in which the member is defined.
- The protected internal access specifier allows a class to hide its member variables and member functions from other class objects and functions, except a child class within the same application.


```
using System;
namespace RectangleApplication {
class Rectangle {
        double length,width;
        public void Acceptdetails() { length = 4.5; width = 3.5; }
        public double GetArea() { return length * width; }
        public void Display() {Console.WriteLine("Area: {0}", GetArea()); }
}
class ExecuteRectangle {
        static void Main(string[] args) {
                Rectangle r = new Rectangle();
                r.Acceptdetails();
                r.Display();
                Console.ReadLine(); } } }
```

Passing Parameters to a Method

- Value parameters : n.swap(a, b);
- Reference parameters

  public void swap(ref int x, ref int y) {----}

  n.swap(ref a, ref b);
- Output parameters

  A return statement can be used for returning only one value from a function. However, using **output parameters**, you can return two values from a function. Output parameters are similar to reference parameters, except that they transfer data out of the method rather than into it.


```
using System;
namespace CalculatorApplication {
class NumberManipulator {
public void getValues(out int x, out int y ) {
        Console.WriteLine("Enter the first value: ");
        x = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Enter the second value: ");
        y = Convert.ToInt32(Console.ReadLine()); }
static void Main(string[] args) {
NumberManipulator n = new NumberManipulator();
int a , b;
n.getValues(out a, out b);
Console.WriteLine("After method call, value of a : {0}", a);
Console.WriteLine("After method call, value of b : {0}", b);
Console.ReadLine(); } } }
```


Arrays

- double[] balance = new double[10]; / / declaration
- double[] balance = { 2340.0, 4523.69, 3421.0}; / / initialisation
- int [] marks = new int[5] { 99, 98, 92, 97, 95}; / / both
- int [] marks = new int[] { 99, 98, 92, 97, 95}; / / omit the size
- int[] score = marks; / / copy
- double salary = balance[9]; / / accessing an array
- foreach (int j in arr) {

  Console.WriteLine("Elements are:", j); }
- int [,] a = int [3,4] = { {0, 1, 2, 3} , {4, 5, 6, 7} , {8, 9, 10, 11}};
- int val = a[2,3]; / / accessing multidimensional array

Jagged Arrays

- A Jagged array is an array of arrays. Not all "rows" of data need to have the same number of elements.
- int[][] scores = new int[2][]{new int[]{92,93,94},new int[]{85,66,87,88}};
- string[][] Members = new string[2][];

Members[0] = new string[]{"Celeste", "Mathuri", "Alex", "Germain"};

Members[1] = new string[]{"Jeremy", "Mathew", "Anselme", "Fred"};

- It is important to know that the pairs of **brackets** indicate "jagged," and the comma in the brackets means "2D".
- Only the **reference** is copied on each function call.

Jagged vs. 2D Array in Performance

2D array code benchmarked: C# // 2D array of 100 x 100 elements.

```
for (int a = 0; a < 100; a++) {
        for (int x = 0; x < 100; x++) {
                int c = a1[a, x]; } }
```

Jagged array code benchmarked: C# // Jagged array of 100 x 100 elements.

```
for (int a = 0; a < 100; a++) {
        for (int x = 0; x < 100; x++) {
                int c = a2[a][x]; } }
```

Results:

2D array looping: 4571 ms

Jagged array looping: 2864 ms [faster]

C# Program to Get a Number and Display the Number with its Reverse

```
int num, reverse = 0;
Console.WriteLine("Enter a Number : ");
 num = int.Parse(Console.ReadLine());
while (num != 0)
{
reverse = reverse * 10;
reverse = reverse + num % 10;
num = num / 10;
}
Console.WriteLine("Reverse of Entered Number is : "+reverse);
Console.ReadLine();
```


Check Whether the Entered Year is a Leap Year or Not

```
namespace Program{
class leapyear {
static void Main(string[] args) {
leapyear obj = new leapyear();
obj.readdata();obj.leap(); }
int y;
public void readdata() {
Console.WriteLine("Enter the Year in Four Digits : ");
y = Convert.ToInt32(Console.ReadLine()); }
public void leap() {
if ((y % 4 == 0 && y % 100 != 0) || (y % 400 == 0)) {
Console.WriteLine("{0} is a Leap Year", y); }
else {Console.WriteLine("{0} is not a Leap Year", y); }
Console.ReadLine(); } } }
```


Display All the Prime Numbers Between 1 to 100

```
namespace PrimeNumber{ class Program {
static void Main(string[] args){
 bool isPrime = true;
Console.WriteLine("Prime Numbers : ");
for (int i = 2; i <= 100; i++) {
for (int j = 2; j <= 100; j++) {
 if (i != j && i % j == 0) {
 isPrime = false; break; }  }
if (isPrime) {
Console.Write("\t" +i); }
isPrime = true; }
Console.ReadKey(); } }}
```

Inheritance

```
using System;
namespace Exercises {
  public class SecondClass    {
     protected int a;
      public SecondClass()       {       a=100;       }
      public void display()    {       Console.WriteLine("a={0}",a);     }
  class derived : SecondClass {
   public void change()
         {          a = a + 20;       }   }   }
```


Regular Expressions

- It is a pattern that could be matched against an input text.
- using System.Text.RegularExpressions;
- **A class,** Regex, handles regular expressions. We specify patterns as string arguments. **Methods** (like Match and Replace) are available.
- Sometimes one match is not enough. Here we use Matches instead of Match: it returns multiple Match objects at once. These are returned in a MatchCollection.


```
String input = "When the above code is compiled and
executed, it produces the following result";
        Console.WriteLine("Starts with c");
        String pattern = @"\bc\S*";
        MatchCollection mc=Regex.Matches(input, pattern);
        foreach (Match i in mc)
          Console.WriteLine(i);
```

\b➔ word boundry
\S➔ non-white-space character


```
Console.WriteLine("Starts with m and ends with e");
        input = "make maze and manage to measure it ";
        pattern = @"\bm\S*e\b";
        MatchCollection m = Regex.Matches(input, pattern);
        foreach (Match i in m)
          Console.WriteLine(i);
```

***➔0 or more times**
+➔1 or more times
{n,m}➔from n to m times

```
Console.WriteLine("Starts with digits");
input = "28989 assssa 123 4555555 234 aas 4werr 567hjj235
878vdfvjdf897";
pattern=@"\b\d*\b";
MatchCollection c = Regex.Matches(input, pattern);
foreach (Match i in c)
        Console.WriteLine(i);
```

\d➔digit


```
Console.WriteLine("Email Validation");
input="v.marees-wari@vit.ac.in";
pattern=@"^([\w\-\.]+)@([\w]+)((\.(\w){2,3})+)$";
      if(Regex.IsMatch(input,pattern))
        Console.WriteLine("Valid Email");
      else
        Console.WriteLine("Invalid Email");
```

\w➔word character [a-z]➔in the a-z range

^➔at start of string or line $➔ at end of string or line

File Operations

```
string f =@"F:\Academic\Web Services Lab\sample.txt";
```

Check:

```
File.Exists(f)
```

Reading a File:

```
string file = File.ReadAllText(f);
Console.WriteLine(file);
```

For Writing:

```
Console.WriteLine("Please enter new content for the file:");
string newContent = Console.ReadLine();
File.WriteAllText(f, newContent);
```


To append the content at EOF:

```
while (newContent != "exit")
{
File.AppendAllText(f, newContent + Environment.NewLine);
                              //add empty line at end of the file
      newContent = Console.ReadLine();
}
```

To delete a file:

```
File.Delete("test.txt");
```

- **Renaming a file**

```
File.Move("test.txt", newFilename);
```

- DirectoryInfo mydir = new DirectoryInfo(@"c:\Windows");
- FileInfo [] f = mydir.GetFiles();
- **Properties:** CreationTime, Directory, Exists, Extension, Name, LastAccessTime,Attributes(Readonly, hidden, system, temporary..)


```
try
{       //block of code to monitor for errors
}
catch (ExceptionType1  identifier)
{       // exception handler for ExceptionType1
}
catch (ExceptionType2 identifier)
{       // exception handler for ExceptionType2
}
finally
{       // block of code to be executed before try block ends
}
```


Keywords

- Java exception handling is managed by via five keywords: **try, catch, throw, throws,** and **finally.**
- Program statements to monitor are contained within a **try** block
- If an exception occurs within the **try** block, it is thrown.
- Code within **catch** block catch the exception and handle it.
- System generated exceptions are automatically thrown by the Java run-time system.
- To manually throw an exception, use the keyword **throw.**
- Any exception that is thrown out of a method must be specified as such by a **throws** clause.


```
using System;
namespace ErrorHandlingApplication {
class DivNumbers {
int result;
DivNumbers() { result = 0; }
public void division(int num1, int num2) {
try { result = num1 / num2; }
catch (DivideByZeroException e) {
Console.WriteLine("Exception caught: {0}", e); }
finally { Console.WriteLine("Result: {0}", result); } }
static void Main(string[] args) {
DivNumbers d = new DivNumbers();
d.division(25, 0); Console.ReadKey(); } } }
```