**C# Language**

Today you will learn

• The .NET Languages

• Variables and Data Types

• Variable Operations

• Object-Based Manipulation

CSE 409 –Advanced Internet Technology

The .NET Languages

• The .NET Framework ships with two core languages that are commonly used for building ASP.NET applications: VB and C#.

• These languages are, to a large degree, functionally equivalent.

CSE 409 –Advanced Internet Technology 2

Variables and Data Types

• Variables can store numbers, text, dates, times, and they can even point to full-fledged objects.

• When you declare a variable, you give it a name, and you specify the type of data it will store.

//Declare a string variable named myName.

string myName;

Note :Comments are descriptive text that is ignored by the compiler. C# comments provides two basic types of comments:

//A single-line C# comment /\* A multiple-line

C# comment\*/

CSE 409 –Advanced Internet Technology 3

Variables and Data Types

**• Every .NET language uses the same variable data types.**

• Different languages may provide slightly different names (for example, a VB Integer is the same as a C# int), but the CLR makes no distinction-in fact, they are just two different names for the same base data type (in this case,it’s System.Int32).

• Because languages share the same core data types, you can easily use objects written in one .NET language in an application written in another .NET language.

**• No data type conversions are required.**

CSE 409 –Advanced Internet Technology 4

Variables and Data Types

CSE 409 –Advanced Internet Technology 5

Assignment and Initializers

**• Once you’ve declared your variable, you can freely assign values to them, as long as these values have the correct data type.**

• Here’s the code that shows this two-step process: string myName; myName = "Matthew ";

**• You can also assign a value to a variable in the same line that you declare it. string myName = "Matthew ";**

CSE 409 –Advanced Internet Technology 6

Assignment and Initializers

**• C# does not allow uninitialized variables.**

• This means the following code will not compile in C#: int number; //Number is uninitialized number = number + 1;//This causes a compile error

• Proper way to write above code: int number=0; //Number now contains 0 number = number + 1; //Number now contains 1

CSE 409 –Advanced Internet Technology 7

Assignment and Initializers

**• If you’re declaring and initializing a variable in a single statement, and if the C# compiler can infer the correct data type based on the value you’re using, you don’t need to specify the data type. Here’s an example: var myString = "This is also a string"**

• myString is created as string, even though the statement doesn’t indicate the string data type.

CSE 409 –Advanced Internet Technology 8

Arrays

• Arrays allow you to store a series of values that have the same data type.

• Each individual value in the array is accessed using one or more index numbers.

• Typically, arrays are laid out contiguously in memory.

• All arrays start at a fixed lower bound of 0.

• Because counting starts at 0, the highest index is actually one less than the number of elements.

CSE 409 –Advanced Internet Technology 9

Arrays

• When you create an array in C#, you simply specify the upper bound:

//Create an array with four strings (from index 0 to index 3). string[] stringArray = new string[4];

//Create a 2 x 4 grid array (with a total of eight integers). int[,] intArray = new int[2, 4];

CSE 409 –Advanced Internet Technology 10

Arrays

**• You can also fill an array with data at the same time that you create it.**

• In this case, you don’t need to explicitly specify the number of elements, because .NET can determine it automatically:

//Create an array with four strings, one for each number from 1 to 4. string[] stringArray = {"1", "2", "3", "4"}

CSE 409 –Advanced Internet Technology 11

Arrays

• The same technique works for multidimensional arrays, except that two sets of curly brackets are required //Create a 4 x 2 array (a grid with four rows and two columns). int[,] intArray = {{1, 2}, {3, 4}, {5, 6}, {7, 8}}

CSE 409 –Advanced Internet Technology 12

Arrays

• To access an element in an array, you specify the corresponding index number in square brackets.

• Array indices are always zero-based. That means myArray[0] accesses the first value in a one dimensional array, myArray[1] accesses the second value, and so on.

• In a two-dimensional array, you need two index numbers:

int[,] intArray = {{1, 2}, {3, 4}, {5, 6}, {7, 8}} //Access the value in row 0 (first row), column 1 (second column). int element; element = intArray[0, 1] // Element is now set to 2.

CSE 409 –Advanced Internet Technology 13

The ArrayList

• In many cases, it’s easier to use a full-fledged collection rather than an array.

• Collections are generally better suited to modern object-oriented programming and are used extensively in ASP.NET.

• The .NET class library provides many types of collection classes, including sorted lists, key-indexed lists (dictionaries), and queues.

• One of the simplest collection classes that .NET provides is the ArrayList, which always allows dynamic resizing.

• Here’s a snippet of C# code that uses an ArrayList:

//Create an ArrayList object.

ArrayList dynamicList = new ArrayList();

CSE 409 –Advanced Internet Technology 14

The ArrayList

• The ArrayList is not strongly typed, so you can add any data type.

dynamicList.Add("one"); dynamicList.Add("two"); dynamicList.Add("three");

/\*Retrieve the first string. Notice that the object must be converted to a string\*/

string item = Convert.ToString(dynamicList[0])

CSE 409 –Advanced Internet Technology 15

Difference between Array and ArrayList

CSE 409 –Advanced Internet Technology 16

Enumerations

**• An enumeration is a group of related constants, each of which is given a descriptive name.**

• Each value in an enumeration corresponds to a preset integer.

• In your code, however, you can refer to an enumerated value by name, which makes your code clearer and helps prevent errors.

CSE 409 –Advanced Internet Technology 17

Enumerations

• Here’s an example of an enumeration that defines different types of users: //Define an enumeration called UserType with three possible values.

**enum UserType {**

Admin, Guest, Invalid }

• Now you can use the UserType enumeration as a special data type that is restricted to one of three possible values.

CSE 409 –Advanced Internet Technology 18

Enumerations

• You assign or compare the enumerated value using the dot notation shown in the following example:

//Create a new value and set it equal to the UserType.Admin constant. UserType newUserType; newUserType = UserType.Admin

• Internally, enumerations are maintained as numbers. In the preceding example, 0 is automatically assigned to Admin, 1 to Guest, and 2 to Invalid.

CSE 409 –Advanced Internet Technology 19

Variable Operations

• You can use all the standard types of variable operations in C#. When working with numbers, you can use various math symbols.

CSE 409 –Advanced Internet Technology 20

Variable Operations

• C# follows the conventional order of operations, performing exponentiation first, followed by multiplication and division and then addition and subtraction.

• You can also control order by grouping subexpressions with parentheses.

int number; number = 4 + 2 \* 3 //number will be 10. number = (4 + 2) \* 3 //number will be 18.

CSE 409 –Advanced Internet Technology 21

Variable Operations

• When dealing with strings, you can use the addition operator (+), to join two strings.

//Join two strings together. fullName = firstName + " " + lastName

• In addition, C# also provides special shorthand assignment operators. Here are a few examples:

//Add 10 to myValue (the same as myValue = myValue + 10). myValue += 10

CSE 409 –Advanced Internet Technology 22

Type Conversions

• Converting information from one data type to another is a fairly common programming task.

• For example, you might retrieve text input for a user that contains the number you want to use for a calculation.

• Conversions are of two types:

**widening and narrowing.**

• Widening conversions always succeed. For example, you can always convert a number into a string, or a 16-bit integer into a 32-bit integer.

CSE 409 –Advanced Internet Technology 23

Type Conversions

• On the other hand, narrowing conversions may or may not succeed, depending on the data.

If you’re converting a 32-bit integer to a 16-bit integer, you could encounter a runtime error if the 32-bit number is larger than the maximum value that can be stored in the 16-bit data type.

• Some strings can’t be converted to numbers.

• A failed narrowing conversion will lead to an unexpected runtime error.

CSE 409 –Advanced Internet Technology 24

Type Conversions

• T

o perform an explicit data type conversion in C#, you need to specify the type in parentheses before the expression you’re converting.

• Here’s how you could rewrite the earlier example with explicit conversions:

int count32= 100; short count16; /\*Explicitly convert your 32-bit number into a 16-bit number. If count32 is too large to fit, .NET will discard some of the info.\*/ count16 = (short)count32;

CSE 409 –Advanced Internet Technology 25

Type Conversions

• Casting cannot be used to convert numbers to string, or vice versa. Instead use the following code:

string countString = "10"; //Convert the string “10” to the numeric value 10 int count = Convert.ToInt32(countString);

//Convert numeric value 10 into the string “10”. countString = Convert.ToString(count);

• Converting string to number will not work if string contains letters or other non-numeric characters.

CSE 409 –Advanced Internet Technology 26

Object-Based Manipulation

• .NET is object-oriented to the core. In fact, even ordinary numeric variables like the ones you’ve seen earlier are really full-fledged objects in disguise.

• This means that common data types have the built-in smarts to handle basic operations.

• For example, all strings are actually complete string objects, with useful methods and properties (such as a Length property that counts the number of characters in the string.)

CSE 409 –Advanced Internet Technology 27

Object-Based Manipulation

• As an example, every type in the .NET class library includes a ToString() method.

• The default implementation of this method returns the class name. In simple variables, a more useful result is returned: the string representation of the given variable.

• The following code snippet demonstrates how to use the ToString() method with an integer:

string myString; int myInteger = 100; //Convert a number to a string. myString will have the contents "100". myString = myInteger.ToString()

CSE 409 –Advanced Internet Technology 28

Object-Based Manipulation

• To understand the previous example, you need to remember that all integer variables are based on the Int32 type in the .NET class library.

• The ToString() method is built into the Int32 type, so it’s available when you use an integer in any language.

CSE 409 –Advanced Internet Technology 29

The String Type

• The following code snippet shows several ways to manipulate a string using the methods in the String type:

String myString = "This is a test string " myString = myString.Trim() //= "This is a test string" myString = myString.Substring(0, 4) // = "This" myString = myString.ToUpper() // = "THIS" myString = myString.Replace("IS", "AT") // = "THAT" int length = myString.Length //= 4

CSE 409 –Advanced Internet Technology 30

The String Type

• The first few statements use built-in methods of the String type, such as Trim(), Substring(), ToUpper() and Replace().

• Each of these methods generates a new string object, which replaces the current contents of the MyString variable.

• The final statement uses the built-in Length property of the String type, which returns an integer that represents the number of characters in the string.

CSE 409 –Advanced Internet Technology 31

The String Type

CSE 409 –Advanced Internet Technology 32

The DateTime and TimeSpan Types

• The DateTime and TimeSpan data types also have built- in methods and properties.

• These class members allow you to perform three useful tasks:

o Extract a part of a DateTime (for example, just the year) or convert a TimeSpan to a specific representation (such as the total number of days or total number of minutes). o Easily perform date and time calculations. o Determine the current date and time and other information (such as the day of the week or whether the date occurs in a leap year).

CSE 409 –Advanced Internet Technology 33

The DateTime and TimeSpan Types

• For example, the following block of code creates a DateTime object, sets it to the current date and time, and adds a number of days. It then creates a string that indicates the year that the new date falls in (for example, 2016).

DateTime myDate = DateTime.Now; myDate = myDate.AddDays(100); string dateString = myDate.Year.ToString();

CSE 409 –Advanced Internet Technology 34

The DateTime and TimeSpan Types

• The DateTime and TimeSpan classes also support the + and – arithmetic operators

DateTime myDate1 = DateTime.Now;

TimeSpan interval = TimeSpan.FromHours(3000); DateTime myDate2 = myDate1 + interval;

//Subtracting one DateTime object from another produces a TimeSpan TimeSpan difference; difference = myDate2 – mydate1

CSE 409 –Advanced Internet Technology 35

The DateTime Members

CSE 409 –Advanced Internet Technology 36

The DateTime Members

CSE 409 –Advanced Internet Technology 37

The TimeSpan Members

CSE 409 –Advanced Internet Technology 38

The Array Type

• All arrays in .NET are actually instances of the Array type

• Length property or the GetLength() method

int[] myArray= {1, 2, 3, 4, 5}; int numberOfElements; numberOfElements = myArray.Length //NumberOfElments = 5

int bound;

//Zero represents the first dimension of an array. bound = myArray.GetUpperBound(0) //Bound = 4

CSE 409 –Advanced Internet Technology 39

The Array Type

• Specific dimension as parameter to GetUpperBound()

//Create a 4 X 2 array (a grid with four rows and two columns). int[,] intArray = {{1, 2}, {3, 4}, {5,6},{7,8}}

int rows = intArray.GetUpperBound(0) + 1 //Rows = 4 int columns = intArray.GetUpperBound(1) + 1 //Columns = 2

CSE 409 –Advanced Internet Technology 40

The Array Type

CSE 409 –Advanced Internet Technology 41

End of lecture

CSE 409 –Advanced Internet Technology 42