



MONASH
University

MONASH
INFORMATION
TECHNOLOGY

FIT5192 Module 2

Internet Applications Development

Lecture 3 Material – C# / ASP.NET Introduction

Lecture Overview

- 1 Introduction
- 2 **C# basics**: data types and operators
- 3 C# Language **Constructs**
- 4 **Major Task** Question

Background

Developed in competition with Java, so some similarities

- Strongly typed
- Object Oriented

Unlike some other languages such as JavaScript

External Resources: MSDN

C# Reference:

<http://msdn.microsoft.com/en-us/library/kx37x362.aspx>

.NET Reference

<http://msdn.microsoft.com/en-us/library/bb400852%28v=vs.100%29.aspx>

Strongly Typed

- Each variable must be **declared** before use.

e.g

```
string memberName = "John Smith";
```

```
int age = 35;
```

Example (part 1):

```
<%@ Page Language="c#" %>
<script runat="server">

    void Page_Load()
    {
        string memberName = "John Smith;
        int age =35;

        Display1.Text = memberName;
        Display2.Text = age.ToString();
    }
</script>
```

Example (part 2):

```
<html>
  <head>
    <title>Creating Variables Example</title>
  </head>
  <body>
    Our new member is:
    <asp:label id="Display1" runat="server" />
    <br />Member's age is:
    <asp:label id="Display2" runat="server" />
  </body>
</html>
```




C# Data type conversion functions

Converting to Strings

C# has 2 built in methods of converting from one datatype to Strings

```
int age = 35;
```

```
string strAge1 = age.ToString();
```

```
string strAge2 = Convert.ToString(age);
```

Convert.ToString() function can handle null values

ToString() can not handle null values.

Comparison Operators

C# Comparison Operators

Similar to Java and C/C++

`==` Equality

`!=` OR `<>` Inequality

`<` Less Than

`>` Greater Than

`<=` Less Than or Equal To

`>=` Greater Than or Equal To

C# Logical Operators

&& And

|| OR

! NOT

Data Types: Strings and Characters

C# Characters and Strings

char data type holds one character
values are enclosed in single quotes.

String data type holds any length of characters
enclosed in double quotes.

```
char initial = 'M';  
string name = "Mouse";
```

string is an Alias for String in C#

Although a String is a reference type

- == and != are defined to:
- compare the values of string objects and not their references
- This is how we would normally want them to behave

C# String manipulation Functions (Part 1)

length	returns the length of a string or string variable
substring (start, length)	extracts length number of characters from string, beginning at position start
IndexOf (character)	returns index number of first occurrence of character within string or -1 if not found
LastIndexOf (character)	returns index number of last occurrence of character within string or -1 if not found
ToUpper	returns uppercase version of string
ToLower	returns lowercase version of string

C# String manipulation Functions (Part 2)

Contains (string)	returns a value indicating whether string occurs within the string
Trim	trims whitespace from start and end of string
TrimStart	trims whitespace from start of string
TrimEnd	trims whitespace from end of string

String manipulation Examples

```
<script runat="server">
    void Page_Load()
    {
        string myString = "the quick brown fox";
        Display1.Text = myString.Length.ToString();
        Display2.Text = myString.Substring(0, 9);
        Display3.Text = myString.IndexOf('o').ToString();
        Display4.Text = myString.LastIndexOf('o').ToString();
        Display5.Text = myString.ToUpper();
        Display6.Text = myString.ToLower();
        Display7.Text = myString.Contains("xx").ToString();
    }
</script>
```

More String manipulation

Splitting and joining strings:

`split(separator, string_to_split)`
returns an array of strings

`join(separator, array_name)`
returns a string of all array elements joined with
separator

Splitting and Joining Strings Example

```
<script runat="server">
    void Page_Load()
    {
        string myDate = "5/8/2011";
        string[] dates = myDate.Split('/');

        Display1.Text = dates[0];
        Display2.Text = dates[1];
        Display3.Text = dates[2];
        Display4.Text = string.Join("/", dates);
    }
</script>
```

Data Types: Numerics

C# arithmetic operators

- + Addition
- Subtraction
- * Multiplication
- / Division
- % Modulus (returns the remainder when one number is divided by another)

C# Arithmetic Example

```
<script runat="server">
```

```
void Page_Load()
```

```
{
```

```
    int num1 = 5;
```

```
    int num2 = 2;
```

```
    Display1.Text = "5 + 2 = " + (num1 + num2).ToString();
```

```
    Display2.Text = "5 - 2 = " + (num1 - num2).ToString();
```

```
    Display3.Text = "5 * 2 = " + (num1 * num2).ToString();
```

```
    Display4.Text = "5 / 2 = " + (num1 / num2).ToString();
```

```
    Display5.Text = "5 % 2 = " + (num1 % num2).ToString();
```

```
} </script>
```


Division of Integers

If we want a Double or Floating point output of an integer division then the data types must be converted

```
int num1 = 5;
```

```
    int num2 = 2;
```

```
Display6.Text = "5 / 2 = " +
```

```
    (Convert.ToDouble(num1) /  
    Convert.ToDouble(num2)).ToString();
```



Other C# Maths functions

```
<script runat="server">
```

```
void Page_Load()
```

```
{
```

```
    Display1.Text = "Ceiling(5.2) = " +  
    Math.Ceiling(5.2);
```

```
    Display2.Text = "Floor(5.2) = " +  
    Math.Floor(5.2);
```

```
    Display3.Text = "Sqrt(144) = " +  
    Math.Sqrt(144);
```

```
    Display4.Text = "Pow(3,3) = " +  
    Math.Pow(3, 3);
```

```
}
```

Data Types: Boolean and bool

Boolean Data Types

Can have only a **true** or **false** value

They can be declared using either **Boolean** or **bool**

bool is an alias for **Boolean**

Boolean Example

```
<script runat="server">  
    void Page_Load()  
    {  
        Boolean result = 5 > 2;  
  
        Display1.Text = "5 > 2 = " + result;  
        Display2.Text = "2 > 5 = " +  
Convert.ToBoolean("2 > 5");  
    }  
</script>
```

Data Types: Dates

Date time examples

```
<script runat="server">
    void Page_Load()
    {
        Display1.Text = "Current Date and Time is " + DateTime.Now;
        Display2.Text = "Short Date = " + DateTime.Now.ToShortDateString();
        Display3.Text = "Short Time = " + DateTime.Now.ToShortTimeString();
        Display4.Text = "Day of Week = " + DateTime.Now.DayOfWeek;
        Display5.Text = "Month = " + DateTime.Now.Month;
        Display6.Text = "Month Name = " + DateTime.Now.ToString("MMMM");
    }
}
```

More Date time examples

```
<script runat="server">
```

```
void Page_Load(){
```

```
    Display1.Text = "Current Date and Time is " + DateTime.Now.ToString("g");
```

```
    Display2.Text = "Current Date is " + DateTime.Now.ToString("d");
```

```
    Display3.Text = "Current Time is " + DateTime.Now.ToString("T");
```

```
    Display4.Text = "Current Time is " + DateTime.Now.ToString("t");
```

```
    Display5.Text = "Current Day is " + DateTime.Now.ToString("dddd");
```

```
    Display6.Text = "Current Month and Year is "+DateTime.Now.ToString("Y");
```

```
    Display7.Text = "Current Month is " + DateTime.Now.ToString("MMMM");
```

```
    Display8.Text = "Current Year is " + DateTime.Now.ToString("yyyy");
```

```
} </script>
```


Data Types: Constants

Constants

Sometimes a value in an application doesn't change:

- e.g Scientific constants:

```
const int AbsoluteZero = -273;
```

The compiler ensures the program doesn't attempt to change the value

Constants conventions

Normally constants are named in capitals:

```
const int ABSOLUTE_ZERO = -273;
```

The `const` keyword is enforced by the compiler,
not the UPPER CASE!

Data types: Arrays

C# Arrays

- **Collections** of, usually related, values
- stored under a single name
- C# does not require a value to be stored in each element
- values do not have to be stored sequentially
- E.G.

```
string[ ] OzStates = new string[6];  
OzStates[0] = "New South Wales";  
OzStates[1] = "South Australia";
```


More C# Arrays

E.G.

```
string[ ] OzStates = new string[6];  
OzStates[0] = "New South Wales";  
OzStates[4] = "South Australia";
```

Or

```
String[ ] OzStates = { "New South Wales",  
    "South Australia",  
    "Tasmania"
```

C# Array Functions

There are many Array functions

- E.G.
- `Array.Sort(OzStates);`



C# code execution sequence

code execution

Branching Statements

Looping Statements

Jumping Statements

Functions



MONASH
University

C# Branching Statements

C# Branching Statements

if...else

- Standard Approach

ternary

- Not recommended for code readability

switch statements

- For large lists of options



if ... else examples

```
void Page_Load()
```

```
{
```

```
    int a = 7;
```

```
    int b = 5;
```

```
    if(a < b) {
```

```
        Display1.Text= "a is less then b";
```

```
    } else {
```

```
        Display1.Text = "a is not less than b";
```

```
    }
```

```
}
```


Ternary operator

Display1.Text = $a < b$? "a is less than b" : "a is not less than b";

- Code is executed depending on whether the test is true or false.
- Not recommended for readable code.



switch .. case statement

```
string choice = "train";
```

```
switch (choice) {
```

```
    case "plane":
```

```
        Display1.Text = "Selected to fly"; break;
```

```
    case "car":
```

```
        Display1.Text = "Selected by road";
```

```
break;
```

```
    case "train":
```

```
        Display1.Text = "Selected by rail"; break;
```

```
    default:
```

```
        Display1.Text = "Staying home"; break;
```

C# Loops

C# Loops

for Loops

while Loops

do ... while Loops

foreach loops



MONASH
University

for Loop Example

.....

```
for(int i = 1; i <= 3; i++) {  
    Display1.Text += "This C# for loop will  
    execute 3 times<br />";  
}
```

.....

while Loop Example

```
int loan = 600;
```

```
while(loan > 0) {  
    Display1.Text += "You still owe $" + loan  
    + "<br />";  
    loan -= 200;  
}
```

```
Display1.Text += "Your loan is now repaid";
```

do ... while Loop Examples

- When Executed at least once

```
int loan = 600;
```

```
do {  
    Display1.Text += "You still owe " + loan + "<br />";  
    loan -= 200;  
} while (loan > 0);
```

```
Display1.Text += "Your loan is now repaid";
```


foreach Loop

Iterates through each item in a container class
e.g. array, that supports the IEnumerable
interface

```
.....  
string[ ] OzStates = { "Victoria", "New South  
Wales",  
    "South Australia", "Tasmania" };
```

```
foreach (string str in OzStates) {  
    Display1.Text += str + "<br />";  
}
```

C# Functions

C# Functions

- Execute **another** named block of code
- Control is **returned** to the calling section of code
- Allow the programmer to **re-usable** code sections

C# Function Signatures

- Parameters types and return data types should be specified

```
return_type functionName([parameter_list...])  
{  
    [statement(s);]  
    [return return_value;]  
}
```

C# Function Examples (Part 1)

```
void Page_Load() {  
    DateTime DueDate;  DateTime CheckOutDate;  
    CheckOutDate = DateTime.Now;  
  
    DueDate = FindDueDate(CheckOutDate);  
    Display1.Text = "Your books are being checked OUT on " +  
        CheckOutDate.ToString("d");  
    Display2.Text = "Your books are DUE BACK on " +  
        DueDate.ToString("d");  
}
```




MONASH
University

C# Function Examples (Part 2)

```
DateTime FindDueDate(DateTime  
    CheckOutDate) {  
    return CheckOutDate.AddDays(30);  
}
```

Major Task Question

Major Task for unit: ASP.NET web site for a fictitious (or real) company

- Due for feedback at **regular intervals**, final submission at end of semester
- The task **feature check list** is in the Design Report Template, linked to Task information in the Assignment section on moodle

Major Tasks on Doubtfire

- Details of Task are on Doubtfire
- Code is a Credit Task, that will be extended to a Distinction Task
- Design Report is Credit Task, that will be extended to a Distinction Task
-
- The Distinction versions of the code and design report will then be extended to High Distinction versions
- Research Report is High Distinction Task

Summary

- 1 **Review** week 2 material
- 2 Introduction
- 3 **C# basics**: data types and operators
- 4 C# Language **Constructs**
- 5 **Major Task** Question

What you will do in the Studio this week

Complete topic 5 exercises

Read ahead ASP examples from topic 6 start
looking at Credit Code Task question

Run using Visual Studio 2013



MONASH
University

Thanks and See you in the
Studio!