C#, LIBRARIES, SOURCE CODE REPOSITORY

- HOW THINGS DONE IN THE C# WAY.
- USEFUL LIBRARIES, APIS.
- SOURCE CODE REPO.

GETS DEFAULT VALUE OF A TYPE.

```
static void GetTypeDefaultValue<T>()
{
    var defaultValue = default(T);
    defaultValue.WriteLine();

    default(int).WriteLine(); // 0
    default(char).WriteLine(); // ''
    default(double).WriteLine(); // 0
    default(bool).WriteLine(); // False

    // null
    //default(string).WriteLine();
    //default(object).WriteLine();
    //default(SimpleClass).WriteLine();
}
```

CHECKS NULL.

```
static void DealWithNull()
{
    string test = null;

    // string.IsNullOrEmpty
    if (string.IsNullOrEmpty(test))
    {
    }

    // string.IsNullOrWhiteSpace
    if (string.IsNullOrWhiteSpace(test))
    {
    }

    // the ?? operator - default value if null.
    var text = test ?? "it is null";

    test.WriteLine();

    // use HasValue to check null for value type variable.
    int? testInt = null;
    if (testInt.HasValue)
    {

    }

    // GetValueOrDefault is very handy   return default value of the type if null.
    testInt.GetValueOrDefault().WriteLine();
}
```

COMPARES THE STRING

```
static void CompareString()
{
    string str1 = "abcd";
    string str2 = "efgh";

    // compare after ToUpperInvariant   ignore case (optimized for ToUpper comparison instead of ToLower.)
    if (str1.ToUpperInvariant().Equals(str2.ToUpperInvariant()))
    {

    }

    // or ignore case
    if (str1.Equals(str2, StringComparison.OrdinalIgnoreCase))
    {

    }
}
```

CAST

```
string ss = "100";
int i = int.Parse(ss);
```

```
string s = "100.09";

double d = Convert.ToDouble(s);
decimal dd = Convert.ToDecimal(s);
string s2 = d.ToString();

string dateTimeinString = "2015-01-01";
DateTime dt = Convert.ToDateTime(dateTimeinString);
```

VAR

```
static void UseVar()
{
    object obj = new SimpleClass();

    SimpleClass casted1 = obj as SimpleClass;
    SimpleClass casted2 = (SimpleClass)obj;

    // try use var as it looks neat and short.
    var casted3 = obj as SimpleClass;
    var casted4 = (SimpleClass)obj;
}
```

PARAMS AS PARAMETER

```
UseParamsAsParameter(1, 2, 3, 4, 5);
```

```
static void UseParamsAsParameter(params int[] numbers)
{
    foreach (var n in numbers)
    {
        Console.WriteLine(n);
    }
}
```

BUILD STRING WITH FORMAT

```
static void BuildStringWithFormat()
{
    // string concate
    string name = null;
    int age = 50;
    bool isHealthy = true;

    // use this to conact the strings, much more readable and safe   null values will show as empty string
    var text3 = string.Format("{0} is {1} and he is healthy:{2}", name, age, isHealthy);
    text3.WriteLine();

    // old way is   hard to read! and will throw error if null values.
    var text4 = name + " is " + age.ToString() + " and he is healthy:" + isHealthy;
    text4.WriteLine();
}
```

HANDLE STRING

```
string path = "C:\\Application\\EmailTemplates"; // You need to put extra escape
string path2 = @"C:\Application\EmailTemplates"; // You don't need to put anything
```

USE INITIALIZER ALIAS

```
static void UsesAliasTypeAndInitializerAlias()
{
    // use this way to initialise properties
    var obj = new MyTestClassAlias()
    {
        Name = "Test",
        Year = 2013
    };

    //old ways
    var obj2 = new MyTestClassAlias();
    obj2.Name = "Test";
    obj.Year = 2013;
}
```

STRING BUILDER

```
static void BuildStringWithStringBuilder()
{
    // string concate in a routine and for large string. e.g. build html
    var stringBuilder = new StringBuilder();
    stringBuilder.Append("hello,");
    stringBuilder.AppendLine("good day"); // added new line after
    stringBuilder.AppendFormat("{0} is {1}", "c#", "cool");
    stringBuilder.WriteLine();
}
```

PRINT ARRAY

```
var array = new [] {"hello","good day",1,2,3};
var text = String.Join(",",array);

Console.WriteLine(text);
```

TRY&CATCH EXCEPTION

```
try
{
    string a = "100.09";
    int b = int.Parse(a);

}
catch(Exception e)
{
    Console.WriteLine(e);
    Console.WriteLine("Catch!");
}
finally
{
    // always executed

    Console.WriteLine("Final!");
}
```

n type	Base type	Description	Example
	Object	Base class for all exceptions.	None (use a derived class of this exception).
ception	Exception	Base class for all runtime generated errors.	None (use a derived class of this exception).
OfRang on	SystemExcept ion	Thrown by the runtime only when an array is indexed improperly.	Indexing an array outside its valid range: `arr[arr.Length+1]`
enceEx	SystemExcept ion	Thrown by the runtime only when a null object is referenced.	`object o = null;` `o.ToString();`
ulationE	SystemExcept ion	Thrown by the runtime only when invalid memory is accessed.	Occurs when interoperating with unmanaged co unsafe managed code, and an invalid pointer is
eration n	SystemExcept ion	Thrown by methods when in an invalid state.	Calling `IEnumerator.GetNext()` after removing from the underlying collection.
lExcept	SystemExcept ion	Base class for all argument exceptions.	None (use a derived class of this exception).
tNullEx	ArgumentExc eption	Thrown by methods that do not allow an argument to be null.	`String s = null;` `"Calculate".IndexOf (s);`
tOutOf ception	ArgumentExc eption	Thrown by methods that verify that arguments are in a given range.	`String s = "string";` `s.Chars[9];`
xceptio	SystemExcept ion	Base class for exceptions that occur or are targeted at environments outside the runtime.	None (use a derived class of this exception).
eption	ExternalExcep tion	Exception encapsulating COM HRESULT information.	Used in COM interop.
tion	ExternalExcep tion	Exception encapsulating Win32 structured exception handling information.	Used in unmanaged code interop.

USE TYPEOF

```
int integerValue = 100;
var integerValueType = integerValue.GetType(); //System.Int32

double doubleValue = 100.01;
var doubleValueType = doubleValue.GetType(); //System.Double

var stringValueType = typeof(string); //System.String
```

USE KEYWORDS AS VARIABLE NAME

```
static void UseKeywordsAsVariableName()
{
    var @string = "test";
    @string.WriteLine();

    var @return = "return value";
    @return.WriteLine();
}
```

USING Statement

- Normally use for I/O and database connection
- It ensures that the object is disposed as soon as it goes out scope otherwise you will have a memory leak (object that is no longer used but still takes up memory)
- It implements IDisposable which means it does not require explicit code
- Example

SERIALISATION

- Converting a object into a format that can be stored
- File, Memory buffer, network connection link
- Most popular serialisation types: XML, Json
- For Client-server communication in web application, Json is more popular then XML because it is a more lightweight
- Example

ENUMS

- Is set of named integer constants.

```
public enum XeroStatus
{
    Draft = 54,
    Submitted = 56,
    Authorised = 57,
    Deleted = 59,
    Voided = 60,
    Confirmed = 1093,
    Error = 1095,
    Paid = 1096
}
```

Generic Collection

- IEnumerable : Base interface that following extend or implement. Read-only, just iterate over the collection.
- ICollection: It Modifies the collection or care about its size but no allow direct accessing elements by index.
- IList: Interface. It allows direct accessing elements by index. Also allows edit, elements
- List: Allows direct accessing, modify, sorted, groupby etc

Example of Generic Collection

```
//IEnumerable example
IEnumerable<int> values = from value in Enumerable.Range(1, 10) select value;

//ICollection example
ICollection<string> names = new List<string>();
names.Add("a");
names.Add("b");
names.Add("c");

//IList example
IList<string> iListNames = new List<string>();
iListNames.Add("d");
iListNames.Add("e");
iListNames.Add("f");

//List Example
List<string> listNames = new List<string>();
listNames.Add("g");
listNames.Add("h");
listNames.Add("i");
```

LINQ

- Language Integrated Query
- Query your database using a query language that is similar to SQL but can be compiled inside a .NET application

```
var element1 = new int [] {1,2,3,4,5,6,7,8,9,10};
var element2 = new int [] {1,2,5,6,9,10,33,44,55};

var resultForIntersect = element1.Intersect(element2);
var resultForIntersectWithWhere = element1.Except(element2);
```

```
var intList = new List<int>();

intList.Add(11);
intList.Add(22);
intList.Add(33);
intList.Add(44);
intList.Add(55);

Console.WriteLine(intList.Average());
Console.WriteLine(intList.Max());
Console.WriteLine(intList.Min());
Console.WriteLine(intList.Any());
Console.WriteLine(intList.Sum());
Console.WriteLine(intList.Count());
Console.WriteLine(intList.Remove(33));

foreach(var i in intList.OrderBy(x => x)){
    Console.WriteLine(i);
```

LINQ

```
var intList = new List<string>();

intList.Add("a");
intList.Add("afweqwef");
intList.Add("b");
intList.Add("bbbbb");
intList.Add("bbbbbbbbb");
intList.Add("c");
intList.Add("d");
intList.Add("f");

var totalA = intList.Where(x=> x == "a" || x == "c");
var containB = intList.Contains("b");
var containZ = intList.Any(a=>a == "f");
var groupByString = intList.GroupBy(a=> a);
var firstElement = intList.First();
var lastElement = intList.Last();
var firstOrDefault = intList.Where(x=> x.Contains("a")).FirstOrDefault();
```

- elements which contains “a” or “c” ?
- order by Alphabet desc?
- check if any elements have string “a” OR “c”
- What is output for “intList.Average()”

LINQ

```
public class Cat
{
    0 references
    public int Id { get; set; }
    0 references
    public string Name { get; set; }
    0 references
    public string Color { get; set; }
}
```

Id	Name	Color
1	A	Blue
2	B	Red
3	C	Green
4	D	Blue
5	E	Blue

- How to retum only Name column?
- How to group by multiple column?
- How to order by multiple column?
- Get the list of cat which its color is blue and order by Id
- Get the list of cat which its name either A or E and group by Color and Id and retum only color
- How to sum up by Id?

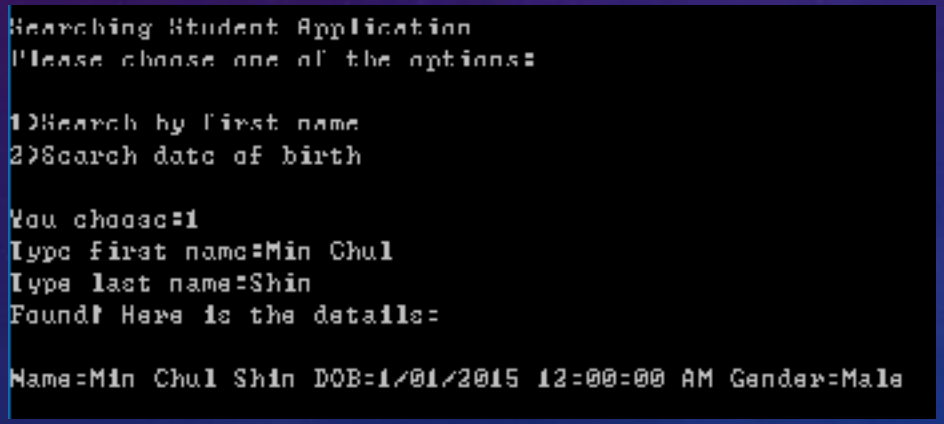
From Model to Linq

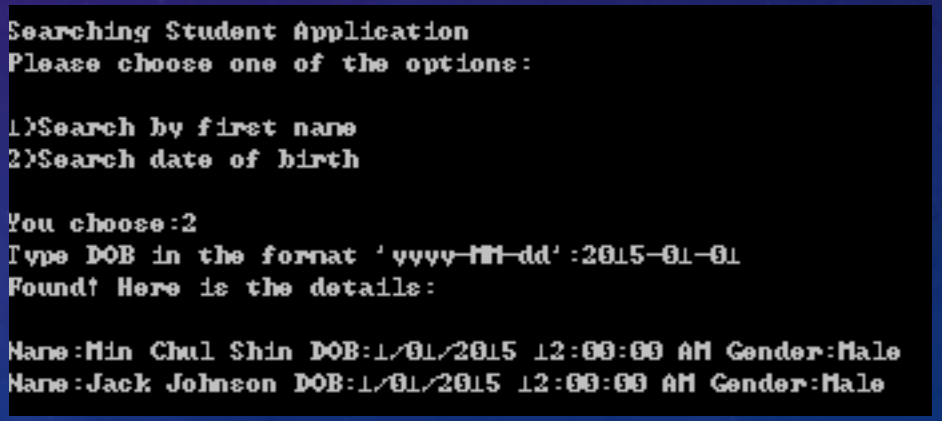
1) Initialise ‘Child’ class with 3 properties:
   Id(int),Name(string),DateOfBirtth(datetime),Gender(string)
2) Create a list which its type is Child and add elements followings:
   - 1, Shin, 1988-01-01, M
   - 2, Jason , 1921-03-03,M
   - 3, Jane, 1933 -04-04, F
   - 4, Lucy, 2010-05-05,F

1) Write the query which returns children whose name contains ‘e’ and born before 2015-01-01
2) Write the query which returns type of gender
3) Find out the children who were born after 2000-02-02 and add +5 to its Id
4) Write the codes if there is child whose id = 3 then return with additional property “Result” with string “Hi”, otherwise string “Bye”
5) Write the query to find out the total children by Gender

C# Exercise

- You need to create a console application which user can search students in the system by either name or date of birth.
- Set student model with properties: Id(int), FirstName(string), LastName(string), DateOfBirth(datetime), Gender(string)
- Create one list with type 'Student' and add following data:
- (1, 'Min Chul', 'Shin', 2011-01-01, Male), (2, 'Amy', 'Park', 1988-03-03, Female)
- Example:





.NET LIBRARIES

- Productivity - not reinvent the wheel
- A measure of practical knowledge.
- Make features achievable.
- Consists of
  - .DOTNET framework libraries.
  - .DOTNET party libraries.
  - 3rd

DOT NET FRAMEWORK LIBRARIES

Provide foundation and functions to solve common stuff.

- E.g.
  System
- System.Collections
- System.Data
- System.Drawing
- System.IO System.Text
- System.Threading
- System.Timers
- System.Web
- System.Web.Services
- System.Windows.Forms
- System.Xml

DOT NET 3RD PARTY LIBRARIES

- Specific feature provided. E.g. pdf
- Productivity & efficiency. Just use it
- Ensure the quality and reliability. They are tested
- Make life much easier. May already solve the problem

PDF DOCUMENTS - 3RD PARTY LIBRARY

- PDFsharp is the Open Source library that easily creates PDF documents from any .NET language.

MORE TO COME - 3RD PARTY LIBRARY

- ASP.NET Form web control libraries.
- ASP.NET MVC web control libraries.
- WPF/Silverlight or Windows form libraries.
- Javascript libraries.
- etc

LOG4NET FOR LOGGING- 3RD PARTY LIBRARY

- Log4net is faster
- Easy to setup
- There are various supported appenders – log delivery methods
  - Email, files, colored console
  - Database, Event log
  - Asp.net trace
  - Udp, Telnet
  - Log to remote server

- **Source Code Repository**
  - Keep track of code changes
  - Ownership of changes
  - Managing merging
  - Keep code safe
  - Build server

TFS – SOURCE CODE SERVER

- Team foundation server – integrated with visual studio
- Source code management
- Team Build Build controller
- Project Management and tracking
- Reporting

TFS– SOURCE CODE SERVER