Mon Tue Wed Thu Fri Sat Sun

Date: 10-02-20

EAD

Lecture : 01

⇒ Konrad Zuse : (1936 – 1943)

Zuse Plankalkul → The very first programming language → 1943

1945 → This idea disappeared.

1940 – 1950 → The programming have started.

Minimal hardware programming language

↳ Write pseudo code.

1972 → Konrad zuse publish its work.

1949 → A new language was born.

⇒ John Machly : (1949)

Give a concept of stored program. 50 times slower than machine code. People then named it as automatic programming language though it was slow.

⇒ John Backus : (1954)

Launch language of speed coding.

Mon Tue Wed Thu Fri Sat Sun

Date:

Give concept of auto increment.

He joined IBM and within a year, he define a new language IBM mathematical formula translating system (FORTRAN). in which he declared 6 premitive integers (i, j, k, l, m, n).

Still FORTRAN is used in many different sector.

Latest version : 2018

⇒ LISP :

AI Based language.

↳ Three pillars of AI (Artificial Intelligence):

- Psychology :
  Think how human think.
- Mathematics : Give mathematical representation of our thinking.
- Linguistic : Convert into programming language.
  To communicate between living and non - living.

Computer understand binary language because there is only 2 current state (ON, OFF)

Mon Tue Wed Thu Fri Sat Sun

Date:

⇒ International Algorithmic Language (ALGOL)

COBOL (Common Business Oriented Language)
Used for business purpose.

⇒ BASIC: (1970 - 1980)
BASIC language for normal layman.
Visual Basic (GUI) → 1990
Visual Basic .Net (Object Oriented) → 2000

⇒ BCPL (1967) By Martin Richard: (University of Cambridge)
Basic combined Programming language.
Simple System langugе
No integers defined
Thompson defined integers in BCPL now known as B language (1ST high level language)
Overcomed the faults of BCPL.
Dennis Ritchie developed C in 1972.

⇒ Alan Kay developed small talk language in 1960:
1ST OOP language (University of UTAH)

Mon Tue Wed Thu Fri Sat Sun

Date:

↳ Reason to introduce OOP:

- Reusability.
- Easy debugging.
- Solution to spaghetti code. (GOTO Statement create a mess that is called spaghetti code).

⇒ C++ : Bjarne Stroustrup

Actual name of C++ is C with classes.
"We don't need to reinvent the wheel"
Resolved issues in C. Concept of Object Oriented.

⇒ Oak : James Gosling developed it.

Name changes and become:
GREEN → DNA → Silk → Neon → Pepper
→ Lyric → NetPraise → Web developer language → Web dancer → Web Spinner

Team love for coffee → JAVA

Developed in 1990.
More work on internet so called NetPraise
Actually use in Embedded System.

Mon Tue Wed Thu Fri Sat Sun

Date: 12-02-20

Lecture : 02

⇒ JAVA : Company : Sun MicroSystem then it buy by Oracle.
1990 → Java develope.
1991 → properly managed its work
1995 → It's compiler.

Actually java appear for embedded system. Because Internet becomes very popular in 1990 so java is mainly use for that.

⇒ Anders Hejlsberg developed C# language.
↓
Chief Architect of Microsoft company.

↳ Why C# developed :

- Compile different programming language in one compiler.
- Overcome the difficulties of C++ and Java.
  In 2000, more newly things are introduced.
- Easy to learn and understand.

⇒ Introduction to .NET framework and C# :

Compiler Name of C# : CLR (Common Language Runtime)
Java Compiler : JVM (Only understand Byte code)

Mon Tue Wed Thu Fri Sat Sun

Date:

Multi-language : If the specification of particular language is present in it then we can utilize this language.

Server side code never view on client side. But we utilize it through hacking but never understand it through hacking.

↳ **CLR :**
- Object decleration, creation and use
- Error and excepting handling
- Data types, language libraries
- Interactive Development Environment (IDE)

⇒ Two parts of CLR :

↳ CTS : Provide base line of language (int, double)

↳ CLS : Provide the support of language.

Java Code → Compile → Byte code understand by JVM again

C# code → Compile → IL (Intermediate language) understand by CLR again.
CLR itself contain a compile that is JLT. (Just in Time).

Mon Tue Wed Thu Fri Sat Sun

Date:

Memory Management : Garbage collector, constructor, destructor.

↳ Built in available libraries are in "Core FCL".

↳ External components / libraries are in "Other FCL components".

C# use .exe file to compile bcz windows is developed by microsoft and it use .exe extension and C# is also developed by microsoft so its extension is also .exe.

- Java compile Byte Code.
- C# compile .exe Code.

Project of C# developed in Window XP, its GUI component is same as Window XP.

Project of C# in Window 2007 → GUI component is same as Window 2007 because Window library is parse in it.

Compile it as : csc abc.cs

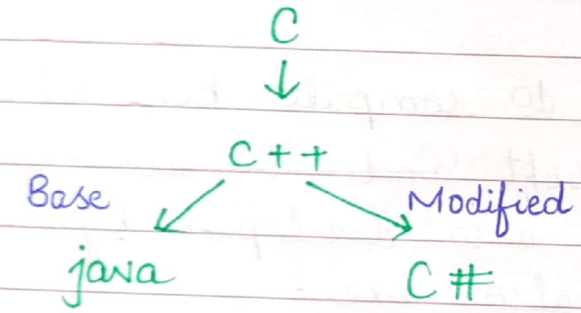
View as : abc

Install
.Net Framework
check → cmd
→ csc

Mon Tue Wed Thu Fri Sat Sun

Date:

```
class first
{
    public static void main (String [ ] arg )
    {
        System. console. Writeline ("Hello");
        System. console. readkey ();
    }
}
```



Java and C# are cousin language.

Framework : It is a platform in which we merge libraries and then utilize these libraries.

Different system combined and work on a framework.

Plateform → framework → Different System → Built in API and libraries.

* Biography of Anders Hejlsberg. (Read)

* All keywords of C# with one line defination (Write on A4 size).

Herbert Schildt C# 4.0 → Textbook

Mon Tue Wed Thu Fri Sat Sun

Date: 17-02-20

Lecture : 03

⇒ Monolithic Application :

Code it and then make .exe file and execute.

Exe file are only executable file. We don't modify it. e.g MS Word

⇒ Component Base App :

Modify the code and utilize it.

DLL : Dynamic Link Library.

Namespace : Just like a folder, contain different things in it. It itself is nothing but contain block of code which can be utilize.

If main exist, then exe file make.

.DLL file don't contain 'main' → no main → No executable file.

Enhance the code to whatever extent but if 'main' is made then code can't be enhanced further.

Make .cs into .dll :

csc /t:library /out:hello.dll helloFile.cs

Mon Tue Wed Thu Fri Sat Sun

Date:

.dll are not executable file, they are used for reuseability. To utilize and to enhance it or just use them by calling.

↳ public is accessible within same software.
↳ protected is accessible within a inherited classes.
↳ private is accessible within a class.

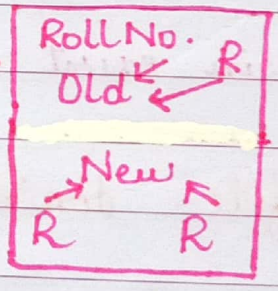
⇒ Static :

*. 'Static' class or not ? In which language?

↳ Function can be 'Static'. It is also called shared object. A static function impact the class. We can directly access or call the function without creating object.

↳ Static data members → Shared Memory location. Initialize only one time / Only one occurence.

*. Static property, Static Constructor.



↳ 'void' → Data type. No argument.

Make .dll to .exe file :

csc /r : h.dll/ out : my.exe b.cs

reference ↳ Multiple .dll can be make into .exe file.

Mon Tue Wed Thu Fri Sat Sun

Date:

↳ Multiple DLL into 1 exe :

csc /r:a.dll,b.dll,c.dll,d.dll /out:output.exe mainPlus.cs

↳ Multiple files into 1 dll :

csc /t:library /out.final.dll myPlusFun.cs mySubFun.cs myMulFun.cs myDivFun.cs *

Namespace component should be same bcz it have to use internally. If name differ it consider it other namespace in other FCL.

* csc /r:final.dll /out:output.exe mainPlus.cs

Mon Tue Wed Thu Fri Sat Sun

Date: 19-02-20

Lecture : 04

⇒ Namespace as Alias :

Visual C# → Console Application

We can assign alias to namespace.

```
using sc = System.Console
sc.WriteLine();
```

One file can contain multiple namespace.
'Using' keyword is used to use namespace.

Nested namespace can also be done. (1 namespace can contain 1 or more namespace).

- object → Keyword Make an object in memory (Special type of object)

Boxing → We boxed because of security.

Boxed var is not a pointer.

```
object c = a        // Boxing
int K = int(c);     // Unboxing
```

Mon Tue Wed Thu Fri Sat Sun

Date:

- Objects point to different location so they are not equal even if they contain same values.

- Variables are equal if they contain same values.

Example: int j = 2
int c = j
c == j // True

object c_1 = j
object c_2 = c
c_1 == c_2 // False

⇒ How we can acquire different values from same function?
Use 'ref' or 'out'.

```
public static void testRef (int a , int b,
    ref int add , out int sub , out int
            mul , out div)
{
    add = a+b;
    sub = a-b;
    mul = a * b;
    div = (float) a/b ;
}
```

Mon Tue Wed Thu Fri Sat Sun

Date:

- Console.WriteLine("{0} + {1} + {2}", input1, input2, add1) Used just for format. Instead of doing concatenation we do this simple way.

↳ ref : Need first value in advance.
Default value required.
If in 'ref' the value is nothing then the default value store otherwise ref variable is overwrite. (It value don't need to compile).

↳ out : Need to compile. Its don't need default value. Its depend on further compilation. If we don't compile then it will give error.

↳ Two points must be done:
'ref' and 'out' is used in function defination.
And it is also use in function call.

⇒ Positional Arguments : The arguments must be in position. To check the parameters in a function is exactly in the required position (We do naming arguments).

Positional Arguments : the arguments must be in position while calling the function.

Mon Tue Wed Thu Fri Sat Sun

Date:

```
public static double calArea (double h, double
                                        w)
{
    return (h * 5) / (w + h);
}

Console.WriteLine (calArea (6.878, 3.5656))
```

```
Console.WriteLine (calArea (h: 6.878, w: 3.5656))

Console.WriteLine (calArea (w: 3.5656, h: 6.878))
```

⇒ Named arguments : The argument don't need to be in position. We should know the name of arguments. And then call the function by passing arguments with name.

⇒ Optional Arguments :

If default values are given then default run.

Sequence matter : LeftSide arguments can't be skip but rightSide arguments can be skip.

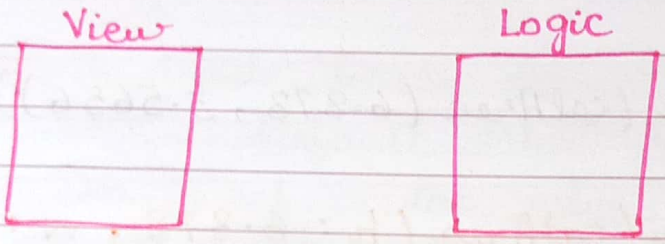
Mon Tue Wed Thu Fri Sat Sun

Date: 24-02-20

Lecture : 05

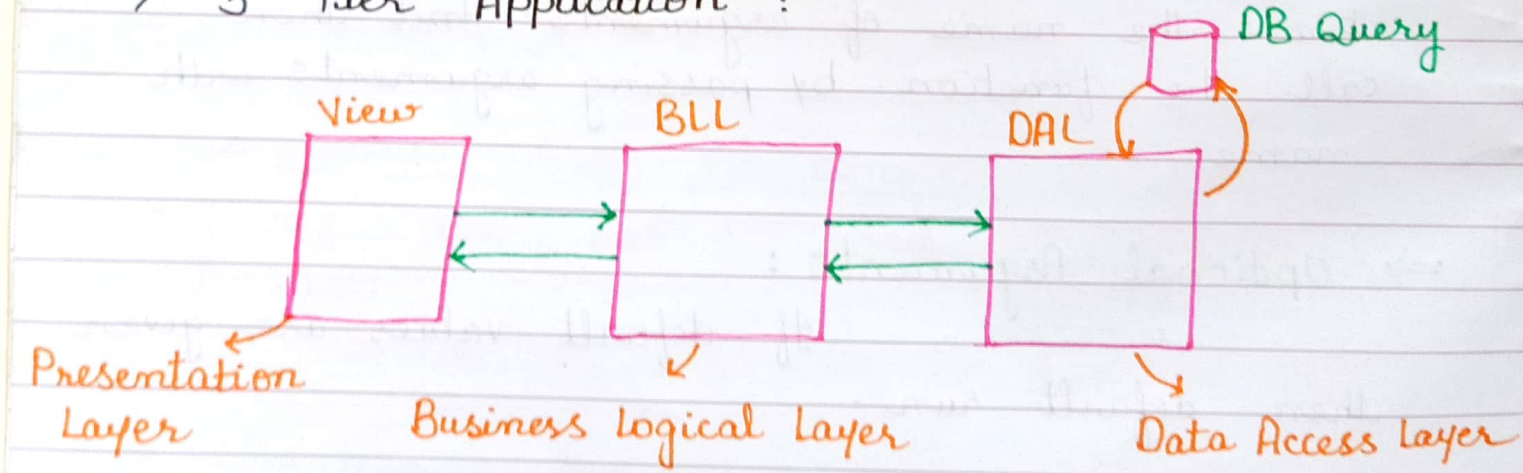
N Tier Application

⇒ 2 Tier Application :



Divide application into two parts then complexity to understand the application is reduce at least 50%.

⇒ 3 Tier Application :



↳ Secure from hackers.
↳ Scalability : Easy to expand.

We create tiers depending upon the complexity of an application.

Mon Tue Wed Thu Fri Sat Sun

Date:

Case Study :

Calculate tax of salary.

↳ Getter and setter in C# :

∴ Class Library (C#)

public string name { set ; get ; }

↳ Bottom-Up-Approach :

When we want to develope Application, first design DB (Design Data Access Layer). DAL → BLL → View

∴ Chap : 1 to 11, 16

using System.IO ; // Files

↳ Create file :

FileStream fout = new FileStream ("data.txt", FileMode.Create);

∴ Refrences → Add → Namespace

↳ Writing object :

StreamWriter sw = new StreamWrite (fout);

↳ Save in file :

sw.WriteLine(b.Name + "," + b.Salary);

sw.Close ( );

fout .close( );

↳ Open file :

FileStream fin = new FileStream ("data.txt", fileMode.Open);

↳ Reading object :

StreamReader str = new StreamReader (fin);

Mon Tue Wed Thu Fri Sat Sun

Date:

↳ Read file:

```
BO K = new BO()
string line = str.Readline();
string[] data = line.Split(',');
K.Name = data[0];
:
return K;
```

Data Source : File Handling + DB + XML.

BLL → Calculate Tax / Logic

DAL → Save in a file / File Handling

All the functions of DAL wrapped in BLL so we don't need to reference DAL in view.

In main only 'PersonView' reference add.

Mon Tue Wed Thu Fri Sat Sun

Date: 26-02-2

Lecture : 06

⇒ Database :

∴ Service Base Database

Identity Specification → True

↓

For auto increment of Primary Key

Seed : Next row

Increment : Auto Increment 1

After create table → Update

* C# → Add → New Item → Data → Service-Based Database → Add

* Database.mdf → Properties → Connection string

Include library :

using System.Data.SqlClient

To establish connection :

Database → properties → connection string → Copy this command.

```
static string con = @" ———— ";
```

↙ accept it as it is, have no meaning.

```
SqlConnection s = new SqlConnection(con);

String query = " insert into Student (Name, Pass)
               values (" + name + ", " + pass + ");
```

Mon Tue Wed Thu Fri Sat Sun

Date:

```
s.Open();
SqlCommand cmd = new SqlCommand(query, s);
cmd.ExecuteNonQuery();
```

It means it does not retrieve something just insert it in table.

```
s.Close();
```

↳ To read data from database :

```
SqlDataReader sdr = cmd.ExecuteReader();
if (sdr.HasRows)
{
    Console.WriteLine("Authenticated");
}
else
{
    Console.WriteLine("Invalid username and pass");
}
```

Sql Injection :

To crash password.

7up' or '2'='2

False condition — True condition

'OR' says that only one condition true and

Mon Tue Wed Thu Fri Sat Sun

Date:

in this '2 = 2' → which is true so it give authenticate username and password.

To overcome Sql injection, Parameterized query is used.

↳ Parameterized Query:

```
Select * from Student where Name = @name
and pass = @pass";
```

Both are same (signature)

```
SqlParameter p1 = new SqlParameter("name",
                                   name);
SqlParameter p2 = new SqlParameter("pass", pass);
```

*• SqlParameter: Execute query first, compile it values as a signature then pass parameter as a value. It takes parameter as value whether we write 'or' / 'and'. All the data considered as a string not a query.

```
SqlCommand cmd =
cmd.Parameters.Add(p1);
```

↳ Solo Learn

C++ JS

Java SQL

*• Student → Id → Identity Specification → HTML

True → Then → Identity Increment → 1 CSS

↳ 6 Chapters Read

Mon Tue Wed Thu Fri Sat Sun

Date: 02-03-20

lecture : 07

⇒ GUI :

Windows 8 → Metro App

Dependency on totally that design of which window we use.

WPF → Window Presentation Formation
Make robust kind of component.

Model Architecture :

Managed code :
Code executed by CLR instead of OS which run in compiler first.

Unmanaged code :
Code executed by OS and not understand by C# compiler. Windows layer

Direct X provide support to run graphic display of direct 3D → Interact with driver.

User 32 → Interact with compiler.

Mon Tue Wed Thu Fri Sat Sun

Date:

Unmanaged layer.

It not only provides the OS GUI but it also facilitates to use external GUI. It provide makeup to show the GUI.

WPF → Backend code (XAML)

∴ C# WPF App

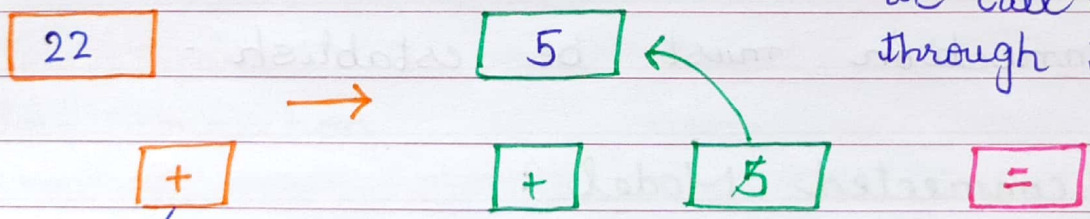
Tool box → Common (Text box) → Button

Change text properties from properties box.

Double click the button to show its program.

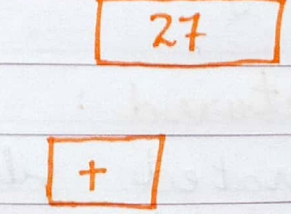
```
<Grid>
<Button Name=
"bt1" >
```

Name must be given so that we call the function through it.



When '+' click store i = 22 and empty the box

When '=' click Store 5 into j then c = i + j and empty the box



Now store sum (c) in that box

Mon Tue Wed Thu Fri Sat Sun

Date: 09-03-20

Lecture : 08

ADO.NET :

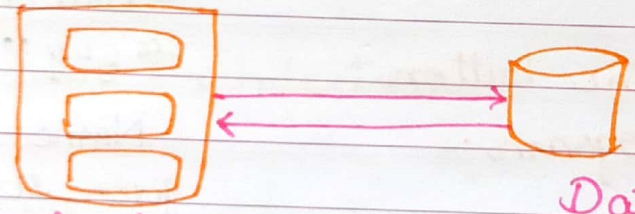
Active X Data object.

Through functions we recognize the database query.

Driver act as bridge

Connected Model :

Attach DB to application.



Connection must be establish.

Disconnected Model :

It can done in multiple ways.

Data Storage :

Structured : Tables, comma seperated/ Space seperated files. It contains excel and CSV files.

Mon Tue Wed Thu Fri Sat Sun

Date:

Hierarchical :

XML

Unstructured Data : Like data on Internet.
Whatsapp picture folder in internal data storage.

Query can't execute on unstructured data.

" " → Must be in search results.

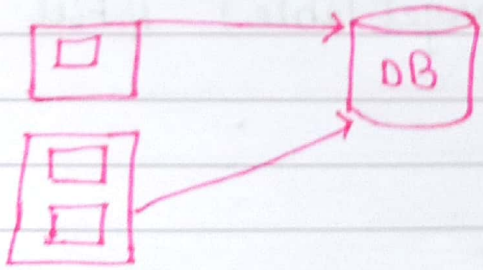
SqlServer - mySql → This 'mySql' can't be appear in search result.

↳ Connected Environment / Connected Model :

1- Open connection
2- Execute command
3- Process rows in reader
4- Close reader
5- Close connection

Advantages :

Work with real data (real db).
Simple Security realization.



Same DB different view -

Mon Tue Wed Thu Fri Sat Sun

Date:

Disadvantages :

Internal connection established.

Disconnected Model :

1. Open the connection
2. Fill the Dataset
3. Close connection
4. Process the Dataset
5. Open Connection
6. Update data source
7. Close connection.

Add Library : using System.Data ;

Copy the path of your previous DB code.

Sql DataAdapter : It provide proper mechanism to create Table.

DataTable : After fetching, where we store our data.

da.Fill (targetTable) // Fill the Dataset

Mon Tue Wed Thu Fri Sat Sun

Date:

DataRow :

Insert new row / Update / Delete.

drow [username] = "Mohsin".

Update :

We should know which row should we want to update. We know the exact row No.

Sql CommandBuilder : Take all the data and store it in DB.

Not directly communicate with data adapter so we need Sql CommandBuilder.

This is need so that we don't give authority to the real data

Mon Tue Wed Thu Fri Sat Sun

Date:

Lecture : 09

11-March-2020

⇒ Delegates :

Delegate is an object that can refer to a method / function. Its purpose is runtime invocation. Used in event handling.

Same concept as Function Pointer.

Declaration :

```
delegate int Sum (int a , int b);
```

Function :

```
static int mySum ( int k , int j)
{
        Console . WriteLine ( k+j );
        return k+j ;
}
```

In Main ( ) function :

- Simple delegate :

```
Sum S = new Sum (mySum);
S (5,6);
Console . Readkey ();
```

Mon Tue Wed Thu Fri Sat Sun

Date:

Delegate and Function return type, No. of parameters and Type of parameters must be same.

1 Delegate can point to multiple functions of same nature.

- Group conversion:

```
Sum s = mySum;
s(5,6);
s = mySub;
s(5,5);
Console.ReadKey();
```

Overlapping exist. "New" Keyword is not added in group conversion.

The method which is refer at last is run first:

Mon Tue Wed Thu Fri Sat Sun

Date:

- Multi-cast delegation :

```
S+ = mySub; // Run all the function at a time. Overlapping not exist. Multiple function can execute. Add method in it

S -= mySum;  // Remove the method from it ('mySum' method remove).
S(6, 7);
```

- Instance Method as delagate :

```
class abc {
        public void display( )
        {
            Console.WriteLine ("Hi");
        }
    }
class Program {
    delagate void d2( ); }
Main ( ) {
    abc a = new abc( ); // class obj
//Make pointer d2 d = a.display; // delegate obj refer to method of class abc.
    d( ); // Do call
}
```

Mon Tue Wed Thu Fri Sat Sun

Date:

```
delegate void divide (int a, int b, int c);

delegate double mul (int a , int b, int c);
```

Function :

```
static void mydivide (int a , int b , int c)
{
    Console.WriteLine ((a/b)/c);
}

static double mymul (int a, int b, int c)
{
    Console.WriteLine (a* b* c);
    return (a*b*c);
}

    Main( )
{
    divide d = new divide (mydivide);
    d (2, 5, 6);
    mul m = new mul (mymul);
    m (2, 5, 6);
```

Mon Tue Wed Thu Fri Sat Sun

Date:

- Anonymous Method : (Unname piece of code).

Anonymous Function

Anonymous Method

```
delegate int add (int a, int b);
Main ():
add h = delegate (int K , int j) {
    Console . WriteLine ( K+j);
    return K+j ;
};
```

OR

```
add h = ( K , j ) ⇒{
    Console . WriteLine ( K+j);
    return K+j ;
};
```

= Compulsory

Lamda Expression

Lamda Statement

- Lamda Expression (Update version of AM).

```
delegate int add (int a, int b);
Main ();
add h =(K, j) ⇒ K + j;
h(5, 5);
```

Left (Parameters) ⇒ Expression

Mon Tue Wed Thu Fri Sat Sun

Date:

⇒ Event : It is basically a notification that an action has occured. It operates delegates and delegates operate function. Event may be multiple.

Event → Delegates → Functions

If one delegate execute multiple functions then one event refer to multiple functions

Mon Tue Wed Thu Fri Sat Sun

Date:

Lecture : 10

13- March-20

Disconnected Model

⇒ Datasets :

No need to have existing database. Create table by yourself.

⇒ Steps :

- Create Tables.
- Create and Add Columns.
- Create and Add rows.
- Create dataset.
- Add table to dataset.
- Create data relations.

1- Create Table :

```
DataTable t1 = new DataTable("University");
```

2- Create and Add columns :

∴ Create columns :

```
DataColumn c1 = new DataColumn("id",
                typeof(int));
DataColumn c2 = new DataColumn("name",
                typeof(string));
```

Mon Tue Wed Thu Fri Sat Sun

Date:

∴ Add columns :

```
t1.Columns.Add(c1);
t1.Columns.Add(c2);
```

3- Create and Add rows :

∴ To access columns :

```
t1["id"]
t1[0]
```

∴ Create rows :

```
DataRow r1 = new DataRow();
DataRow r2 = new DataRow();
```

∴ Add rows :

```
t1.RowsAdd(r1);
t1.RowsAdd(r2);
```

```
r1[id] = 1;
r1["name"] = "Simra";
```

```
c1.AutoIncrement = True
c1.AutoIncrementSeed = 1;
c1.AutoIncrementStep = 1;
t1.PrimaryKey = new DataColumn[] {c1};
```

Identity Increment : 1 → Primary key automatically increment by 1.

Identity Seed : 1 → Move to next row.

Mon Tue Wed Thu Fri Sat Sun

Date:

4- Create dataset :

```
DataSet ds = new DataSet();
```

5- Add table to dataset : (Add two tables
One → Which has primary key.
2nd → Which has foreign key).

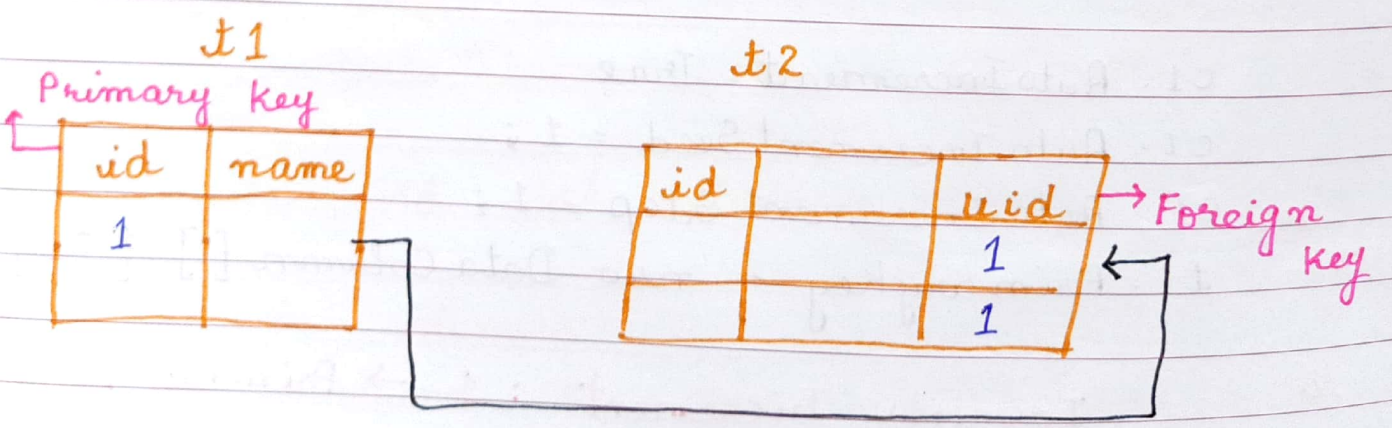
```
ds.Tables.Add(t1);
ds.Tables.Add(t2);
```

6- Create data relations.

```
DataRelation dr = new DataRelation
("R", t1["id"], t2["UId"]);
```

"R" → Relation Name
t1["id"] → Mention primary key
t2["UId"] → Foreign key

```
ds.Relations.Add(dr);
```



Data Row

Mon Tue Wed Thu Fri Sat Sun

Date:

To get child Row:

```
foreach (DataRow rows in t1.Rows)          == Primary key
{
    DataRow child[ ] = rows.GetChildRows("R");
for each (DataRow rowChild in child)        Relation Name
{
    Console.WriteLine(rowChild["Name"]);
    :
    :
}
}
```

Mon Tue Wed Thu Fri Sat Sun

Date: 07-04-20

Lecture : 11

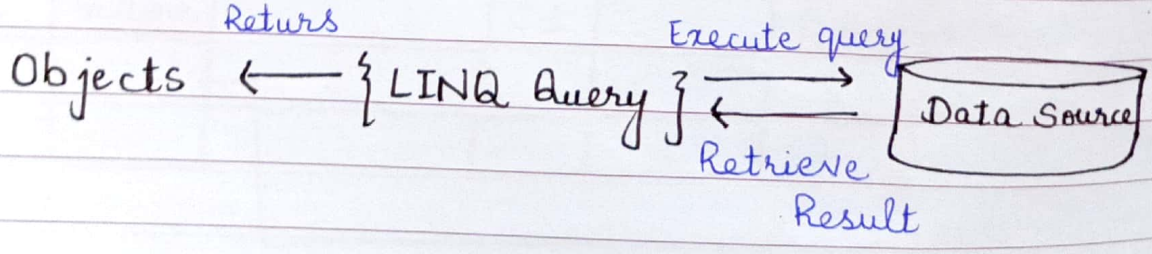
07-04-2020

→ Language - Integrated Query (LINQ):

It was in .NET framework 3.5. By this method we can retreive data from multiple data sources. Data sources meant for any type of file handling, data structures, arrays, linked list etc. To retrieval from the data there is mechanism that's why it is called LINQ.

In other words, it can be called as language within a language.

The C# is a language in which there is another language that is used to retreive the data. In C#, within this language we can utilize the query. No need to use 3rd party software.



Mon Tue Wed Thu Fri Sat Sun

Date:

↳ Add Library :

```
using System.Linq;
```

↳ Syntax of LINQ :

```
var result = from s in strList
             where s.Contains("Tutorials")
             select s;
```

Result variable

Range variable

Sequence (IEnumerable or IQueryable collection)

Conditional expression.

↳ Advantages of LINQ :

From Slides.

It is a uniform way of retreiving data from multiple data sources.

Mon Tue Wed Thu Fri Sat Sun

Date: 09-04-20

Lecture : 12

09- April - 2020

⇒ LINQ - JOIN :

A LINQ Join keyword is used to combine rows from two or more tables, based on a common field between them.

Syntax :

```
Var inStockList = from item in items
                  join entry in statusList
                  on item.ItemID equals
                     item.ItemID
                  select new Temp(item.Name,
                     entry.InStock);
```

Join two different data sources on the behalf of equality in ItemID.

Also use without creating 'Temp' class object.

```
Select new { myName = item.Name,
             Stock = entry.Instock };
```

Mon Tue Wed Thu Fri Sat Sun

Date:

⇒ Generics :

- Generics introduced in C# 2.0.
- Generics allow you to write a class or method that can work with any data type.

** placeholder : <T> → Versitile data type that can accept each kind of data type.

- Generics allow you to define a class with placeholders for the type of its fields, methods, parameters etc.

- Generic replace these place holders with some specific type at compile time.

```
public static bool Check<T>(T a, T b)
{
...
...
}
```

Which type of data is a can be recogonize at compile time.

Example. Check (10, 20);

- It helps you to maximize code reuse, type safety (Give no error even different data types are given) and performance (only one method; less coding)

Mon Tue Wed Thu Fri Sat Sun

Date:

- You can create your own generic interfaces, classes, methods, events and delegates.

- You may get information on the types used in generic data type at run-time.

typeof(T)

- A generic class or method can be defined using angle brackets ( ).

↳ Generics can be applied to the following.

- Interface
- Abstract class
- Class
- Method
- Static Method
- Property
- Event
- Delegates
- Operator

↳ Advantages of Generic:

- Increase the reusability of the code.
- Generic has a performance advantage because it removes the possibilities of boxing and unboxing.

```
public static bool Check(object a, object b)
{ ... }
```

Example: Check(10, 20)

Mon Tue Wed Thu Fri Sat Sun

Date:

Here is : 10 , 20 value type is convert into reference type (object) → Boxing which is not fast for the performance of application.

From generic , we can remove boxing / unboxing .

⇒ Generic Method :

- Generic methods process values whose data types are known only when accessing the variables that store these values.
- A generic method is declared with the generic type parameter list enclosed within angular brackets.

<T> → T is generic type parameter

List → Give multiple <T, S>

generic type parameter ↑

- Defining methods with type parameters allow you to call the method with different type every time.
- You can declare a generic method within generic or non-generic class declarations.

↳ Generic methods can be declared with these keywords :

- Virtual
- Override
- Abstract (No body / Implementation)

Mon Tue Wed Thu Fri Sat Sun

Date:

Lecture : 13

14-April-2020

⇒ LINQ to SQL:

1) LINQ to SQL with mapping :

- Create database table. 'Students' Table.
- Add library : using System.Data.Linq
  using System.Data.Linq.Mapping.
- Add refrence for adding library.
  System.Data.Linq
- Make class of DB

```
[Table (Name = "Students")] // Mapping

class Student
{   // Unique key so that's why Primary key.
    [Column (Name = "Id", IsDbGenerated = true,
Auto generated   // IsPrimary key = true)]

    Public int Id { get ; set ; } // getter setter

    [Column (Name = "Name")] // Column Names

    Public string Name { get ; set ; }

    [Column (Name = "Semester")]
```

Mon Tue Wed Thu Fri Sat Sun

Date:

```
        public string Semester { get ; set ;}
}
```

Data Context : The dataContext is the source of all entities mapped over a database connection.

We use dataContext to pass connectionString We use it to set path (connection string).

To map connection string :

Use colon to inherit 'DataContext' class (Pre - defined class)

We have to set the path in the constructor of 'DataContext' but 'DataContext' is already define. So, We use 'base' keyword to set the path in its constructor.

base : Whatever is passed in the base, its pass it to the constructor of base class 'DataContext'.

```
class data Link : DataContext
{
  public data Link():
        base (@" Connection String here")
  {
  }
```

Take string as it is.

Mon Tue Wed Thu Fri Sat Sun

Date:

```
    public Table<student> students;
}

class Program
{
    static void Main(string[] args)
    {
        dataLink dl = new dataLink();
        var query = from n in dl.students
                    select n;
        foreach(var i in query)
        {
            Console.WriteLine(i.Name + " " + i.Semester);
        }
        Console.ReadKey();

    // Insert data in database:

        student s = new student();
        s.Name = "Simra";
        s.Semester = "1";
        dl.students.InsertOnSubmit(s);
        dl.SubmitChanges();
```

Mon Tue Wed Thu Fri Sat Sun

Date:

2) ⇒ LINQ Query Syntax :

There are two basic way to write LINQ query :

1) Query Syntax or Query Expression Syntax :

```
var result = from s in stringList
             where s.Contains("Tutorials"),
             select s;
```

Conditions

2) Method Syntax or Method Extension Syntax or Fluent :

```
// string collection :
IList<string> stringList = new List<string>
                           (){
                               "C# Tutorials",
                               "VB.NET Tutorials",
                               "JAVA",
                               "Learn C++"
                           };
// LINQ query Syntax :
var result = stringList.Where(s => s.Contains
                              ("Tutorials"));
```

↑ Extension Method

↓ Lambda expression

Mon Tue Wed Thu Fri Sat Sun

Date:

```
var teenAgerStudents = studentList.Where
        (s => s.Age > 12 && s.Age < 20).
            ToList <Student>();
```

// Update the data :

```
s = dl.students.First (a => a.Semester.Equals
                ("1"));
```

- First method
- Last method

OR

```
var queryUpdate = from k in dl.students
                  where k.Semester.Equals("1")
                  select k;
s = queryUpdate.First();

s.Semester = "9";
dl.SubmitChanges();
```

// Delete the data :

```
s = dl.students.First (a => a.Semester.Equals
                ("5"));
dl.Students.DeleteOnSubmit(s);
dl.SubmitChanges();
```

Mon Tue Wed Thu Fri Sat Sun

Date:

3) ↳ LINQ to SQL without using Mapping Concept :

Use drag and drop option LINQ to SQL :

C# → Add New Item → Data → LINQ to SQL classes

↓

extension → .dbml

Create table and then drag the table into .dbml file.

```
static void Main (string[] args)
{
    DataClasses1 DataContext d = new Data
                    Classes1 DataContext ();

    var query = from s in d.Students,
                Select s;
    Console.WriteLine ("SQL query :" +
        d.GetCommand (query).Command
                                    Text);
    foreach (var i in query)
    {
        Console.WriteLine (i.Name);
    }
    // No need to do mapping
    Console.ReadKey ();
}
```

Mon Tue Wed Thu Fri Sat Sun

Date:

Lecture : 14

16 - April - 2020

⇒ Basic requirements to build a website :

HTML → Must → Body

CSS → Beauty

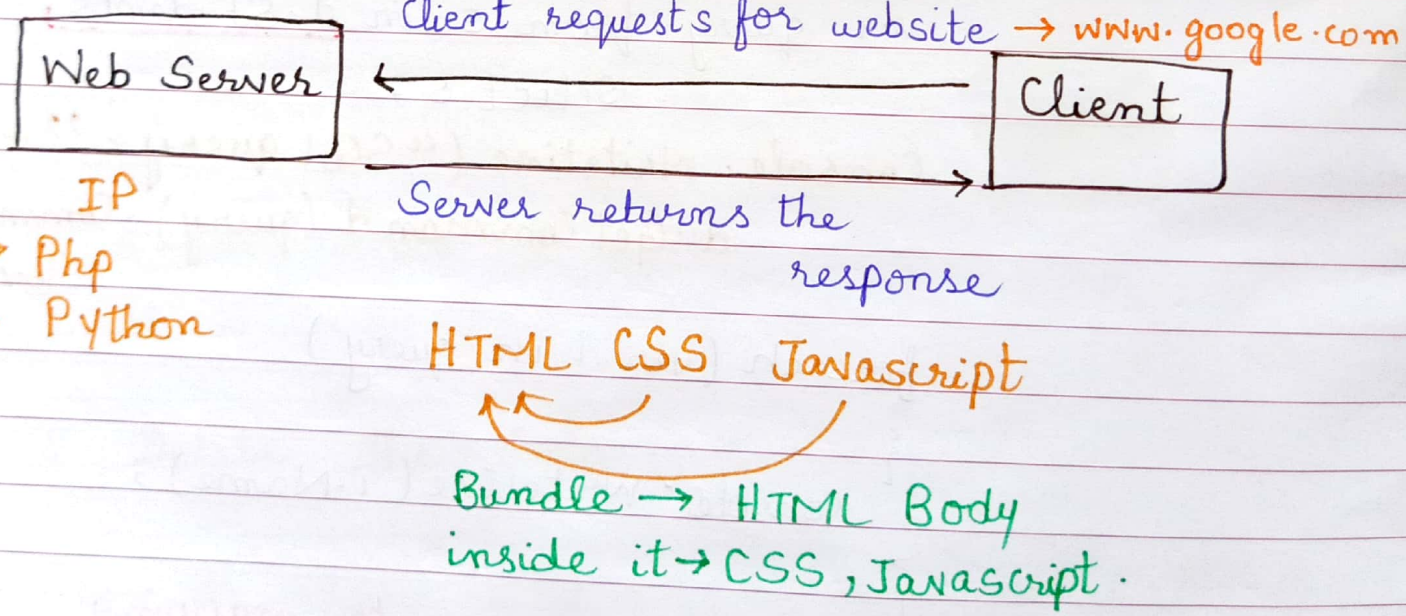
Javascript → Brain / Logics → Client side scripting.

Example : In car :

Metallic body : HTML

Color, design, decors : CSS

Engine, Accelerator / Break / Clutch : Javascript



Mon Tue Wed Thu Fri Sat Sun

Date:

↳ HTML :

```
<! DOCTYPE html >
<html >
 <head >
     <title > </ title >
 </head >
 <body >
     <p> </p >
 </ body>
</html >
```

↳ CSS :

• Internal Style :

```
<head >
    <style >
     p {
         color : white ;
         background : green ;
       }
    </style >
</head >
```

• Inline Style :

```
<body> <p style = "color: blue;" > CSS is Awesome </p >
</body >
```

Mon Tue Wed Thu Fri Sat Sun

Date:

- External Style Sheet :

<link rel = "stylesheet" href = "./style-css" >

Lecture : 15

21- April - 2020

⇒ ASP. NET MVC (Model - View - Controller) :

↳ How to configure ?

- Create a new project
- Select 'ASP. Net Web Application (.Net Framework)' for C#.
- Give the name and create 'MVC'.

Model → That interlink with database present in the form of model.

View → Display

Controller → That control the things.

The standardized way used to develope any application that divide their design, DB classes, Logic (routing, controlling).

↳ Present in a good way.

↳ Easy to code.

Mon Tue Wed Thu Fri Sat Sun

Date:

In MVC, we divide our application into three parts :

Design → View
Database → Model
Routing phases → Controller

(MVC) It used Razor syntax.
↳ File extension used for razor syntax is .cshtml

Home → [@]About.cshtml CSharphtml
↓
Razor Syntax

In cshtml → we are able to do server side programming.

^ → IIS Express → Server → Share port No. where web Application run.

If server is running, then it work otherwise by hitting it URL it can't work.
↳ Stop it in visual studio then server stop running.

Mon Tue Wed Thu Fri Sat Sun

Date:

In Views there exist a folder with a name 'Home' and there must exist a folder (controller) with the same name in controllers. (HomeController)

Add controller :

Right click controller → Add → MVC 5 Controller → Add → Abcd Controller.

In Views :

Right click Abcd → Add new Item → Web → MVC 5 View Page (Razor) → index.cshtml

↓

The same name exist in 'ActionResult' method in AbcdController.cs

Type URL → https://localhost:44333/Abcd/Index

↓

To display output.

* Controller has extension of .cs
* View has extension of .cshtml and the symbol @ which represent view of razor syntax.

Mon Tue Wed Thu Fri Sat Sun

Date:

- **Shared Folder** → Folder which is share to everyone.

  Like → Facebook layout (design) is same for everyone.

  Same design is called layout.

  Layout.cshtml

↳ Layout.cshtml :

How the layout will be?

@ Html.ActionLink("Home", "Index", "Abcd")

To show — ActionResult — Controller

@ RenderBody()

Run the pages that are in this body. (To show the views)

↳ Error.cshtml :

Error page

↳ ViewStart.cstml :

The first request is goes to viewStart → It tell which layout you follow.

Trigger multiple layout.

- **Scripts:** Javascript / JQuery file.
- **Content:** Bootstrap / css / images files.

Mon Tue Wed Thu Fri Sat Sun

Date:

- App-Start : Routing configuration files.
- App-Data : DB in it and DB classes in model.

Means
@ → Server side programming possible.

lecture : 16 23-April-2020

⇒ ASP.NET MVC With HTML Template :

Take template of any website.

- Convert into ASP.Net :

↳ First delete Index.cshtml
↳ Copy all the pages in 'Home' folder.
↳ Rename all the pages from .html into .cshtml.
↳ Make all the functions of the same name as pages (view) in Home Controller.cs.
↳ Copy the images, css and js file by clicking the project and paste.
↳ Use tilt sign '~/' before the css, images and js folder which was copy last time only in html.
It means from project, start from home

Mon Tue Wed Thu Fri Sat Sun

Date:

then CS, js and images folder.
So use before ('~/') wherever images, CS jss written in every pages.

↳ Delete everything except 'render body'.
-layout.cshtml.

↳ Header, footer are written in every pages (same) So we need to do it in generic way so its run in dynamic way. (Only internal body change the other template is same for every pages) → Separate the header, footer and head and paste in in a Shared folder (Add → MVC → MVC 5 partial page (Razor).

↓

Mention that partial page in -Layout.cshtml
@Html.Partial("-Head")

In -Layout.cshtml all those tags are copy which repeat in every pages.

∴ Video → Follow all steps from video lecture.

Mon Tue Wed Thu Fri Sat Sun

Date: 28-04-

lecture : 17

28-April-2020

⇒ ASP.NET MVC (Model using LINQ to SQL Classes):

↳ Create a DB → App Data → Add → New Item → SQL Server Database.

↳ Open the database → Add new Table → Add LINQ to SQL Classes → Drag and drop the table here → Save it.

↳ In 'Contact Page' make a form.

Use 'get' method so that developer view his data on url.

↳ Make a table to receive the data and

1) ✱ 'foreach' loop to show the data after traversing.

↳ DataClasses1 DataContext dc = new DataClasses1....

Go to 'HomeController' → In ActionResult of Contact write a line:

All the record pass in view ← return View(dc.Projects.ToList());

Table name (Add s by default).

↳ Write using 'WebApplication3.Models'. in 'HomeController.cs'

Mon Tue Wed Thu Fri Sat Sun

Date:

We make 'Models' classes to access all the data from database.

Model create a bridge through which working of database can be done in one class.

1) @ foreach (var s in Model) 'Model' has all the things that provide by controller at the backend.

After viewing, make two button through which the data is edit and Delete. (Edit V[illegible] Page).

To 'edit' and 'Delete' we need a that specific entry Id.

↳ Now Go the 'Home Controller' write 'Add()' function to Insert data in table.

After inserting the data redirect it again to 'Contact page'.

```
return RedirectToAction("Contact");
```

↳ Write 'Delete' function that receive Id and delete it by using query Method.

```
var s = dc.Projects.First(x => x.Id == id);
dc.Projects.DeleteOnSubmit(s);
dc.SubmitChang( );
return RedirectToAction("Contact");
```

Mon Tue Wed Thu Fri Sat Sun

Date:

↳ Now write the 'Edit' function which first show the data stored already and then give a option to edit.

Need to make Edit View (EditV) page. Write line in this view :

@model WebApplication3.Models.Project

↙

Strongly Type View → To access the DB (Project table) data in this page.

Access the name and project by using 'Id' that pass in Edit function.

↳ Now make 'EditOK' function to change the edit data.

The data appear in Request after hitting the form

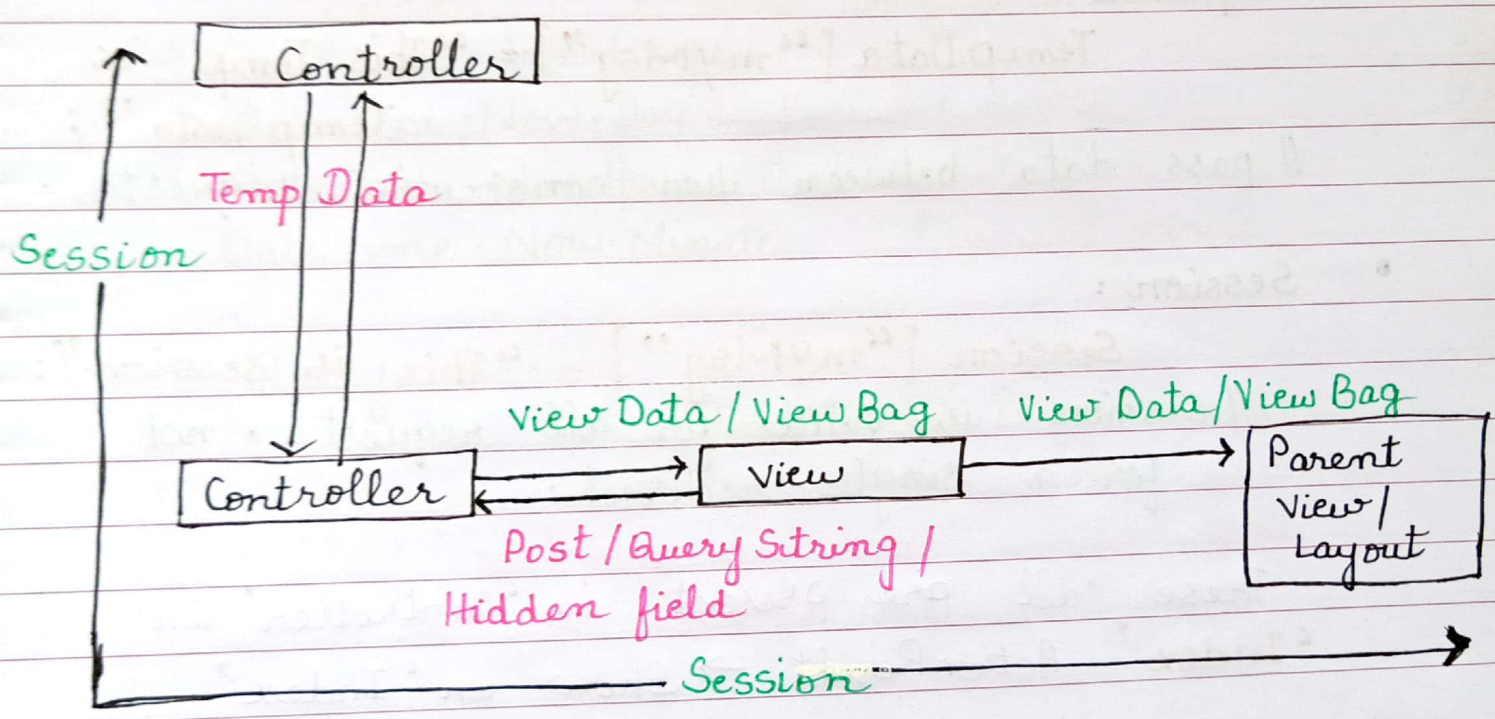
a.Name = Request["name"];

Mon Tue Wed Thu Fri Sat Sun

Date:

Lecture : 18 30-April-2020

⇒ ASP.NET MVC → (View Data, View Bag, TempData, Session, Get/Post Method and Query String):



↳ Syntax :

- ViewBag :

ViewBag.myMsg = "This is ViewBag".
(myMsg: variable)

C# 4.0, Dynamic Type property. Mention it using dot.

Mon Tue Wed Thu Fri Sat Sun

Date:

- View Data :

```
ViewData["myMsg"] = "This is ViewData";
// C# 3.5, contain Key-value pairs that
    must be string.
```

Both 'ViewBag', 'ViewData' are used for data carry.

- Temp Data :

```
TempData["myMsg"] = "This is
                    Temp Data";
// pass data between two consecutive requests.
```

- Session :

```
Session["myMsg"] = "This is session";
// Session is valid for all requests, not
    for a single redirect.
```

These all are present in 'Controller' in 'Index' ActionResult → Show in 'Index'

```
return RedirectToAction("About");
```

↓

After redirect it to 'About' page it only show. tempData and session.

ViewBag and ViewData only used to moved from controller to its view.

Mon Tue Wed Thu Fri Sat Sun

Date:

When redirect from Index to About Page it take 'TempData' with it bcz of redirection But if we go directly to About page it didn't take 'TempData' with it (Because redirection can't be done).

∴ Session is present everywhere.

↳ Built-in-functions:

```
DataTime.Now.Day      // current day
DateTime.Now.Hour     // current hour
DateTime.Now.Minute   // current minute
```

- Query String:

?text1=23 & text2=3

parameters

'Post' method is more secure than 'get' method.

↳ Add Views:

Right Click Views → Add → New Folder (myN) → Right Click myN → Add → View (myView).

↳ Same Name controller add:

Right Click Controller → Add → Controller (myNController)

```
public ActionResult myView()
{
}
```

localhost:1293/myN/myView

Mon Tue Wed Thu Fri Sat Sun

Date:

⇒ Built-In HTML Helpers :

↳ Strongly Typed HTML helpers :

Model (Property) + View (HTML) → Strongly Type View

How to use ?

It use Lambda expressions.

@ Html.TextBoxFor(model ⇒ model.property)

* How to pass data from controller to View using Strongly binding :

- Add controller 'HomeController'
- Add Model → Class 'Employee'.
- Take three data members in class 'Name', Address, 'IsEmployee' as Model property.
- Add view 'uncheck layout page'.
- Write '@using StronglyTypeHelpers.Models', '@model.Employee' in Index.cshtml
- Write @Html.TextBoxFor(x ⇒ x.Name) in body.

  ↙ Strongly Type View

  ↑ Bind Model properties here.

- To show data in View pass the model object (Employee object) in View.

  return View(emp)

Mon Tue Wed Thu Fri Sat Sun

Date:

* How to pass data from view to controller :

Write @using (Html.BeginForm("myform")
∴ if 'using' is not use then need to write endForm.

{
@Html.TextAreaFor(x ⇒ x.Name)
.....
.....
}

Now overwrite the index method by passing an Employee object in 'HomeController'
↓ Before it write [HttpPost]

lecture : 19 05-05-2020

⇒ XML and JSON :

↳ XML :

- XML stands for extensible Markup Language.
- XML is a markup language much like HTML
- Utilize it to communicate between website to website.
- XML was designed to store and transport data. (File Handling).
- XML was designed to be self-descriptive.
- XML is a WC3 Recommendation.

Mon Tue Wed Thu Fri Sat Sun

Date:

Standerized format through which one website utilize the data of other website.

Self made tag can be verify. HTML tag can be also used (there is no restriction of using HTML tag)

```
<note>
  <to> Tove </to>
  <from> Jani </from>
  <heading> Reminder </heading>
  <body> Don't forget me </body>
</note>
```

* Not defined in XML standard.

↓

It has sender information.
It has receiver information.
It has heading.
It has a message body.

But still, the XML above does not DO anything. XML is just information wrapped in tags.

Difference between XML and HTML:

XML was designed to carry data - with focus on what data is.	HTML was designed to display data - with focus on how data looks

Mon Tue Wed Thu Fri Sat Sun

Date:

XML tags are not pre-defined like HTML tags are.

XTML is extensible.

Most XML application will work as expected even if new data is added (or removed).

```
<note>
    <date> 2015-09-01 </date>
    <hour> 08:30 </hour>
    <to> Tove </to>
    .....
</note>
```

∴ Older version of the application can still work.

XML simplifies things:

It simplifies:

Data sharing
Data transport
Platform changes (Send XML from java to C#).
Data availability

W3C → World Wide Web Consortium.
Develop to follow some standard.
Check the error from the website.

Mon Tue Wed Thu Fri Sat Sun

Date:

W3C is currently led by Tim-Berners-Lee

↓

Inventor of world wide web

↳ JSON :

- JSON : JavaScript Object Notation
- JSON is a syntax for storing and exchanging data.
- JSON is text, written with javascript object notation

*. Sending Data :

```
var myObj = { name : "John", age : 31,
              city : "New York" };
var myJSON = JSON.stringify(myObj);
window.location = "demo-json.php?x" + myJSON;
```

*. Receiving Data :

```
var myJSON = '{ "name" : "John", "age" : 31,
               "city" : "New York" }';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML =
                                    myObj.name;
```

- JSON is a lightweight data-interchange format
- Self describing
- Language independent

Mon Tue Wed Thu Fri Sat Sun

Date:

* JSON example :

```
{ "employees" : [
    { "firstname" : "John" , "lastname" : "Doe" } ,
    { "firstname" : "Anna" , "lastname" : "Smith" }
] }
```

* XML example :

```
<employees>
    <employee>
        <firstname> John </FirstName> <lastname>
    <employee>                          Doe </LastName>
</employees>
```

↳ JSON is like XML Because :

Both are :

- Self describing (human readable)
- Hierarchical (values within values)
- Can be passed and used by lots of programming languages.
- Can be fetched with an XMLHttpRequest.

↳ JSON is unlike XML Because :

- JSON don't use end tag
- It is shorter, quicker to read and write
- Can use arrays.

Mon Tue Wed Thu Fri Sat Sun

Date:

XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

JSON is Better than XML because :
- XML is much more difficult to parse than JSON.
- JSON is parsed into a ready-to-use Javascript object.

Website used to verify JSON : jsonviewer.stack.hu

⇒ Built-In HTML Helpers :

↳ Templated HTML helpers :

- Used for data display and input.
- It generate HTML automatically as per model property.

  Means : Calendar from Date, Number from int, Checkbox from bool etc.
- Can generate HTML for complete model with a single tag.

◉ Types of Templated HTML helper :

- Display

  @HTML.Display(), @HTML.DisplayFor(),

Mon Tue Wed Thu Fri Sat Sun

Date:

@HTML.DisplayName(), @HTML.DisplayNameFor(), @HTML.DisplayText(), @HTML.DisplayTextFor(), @HTML.DisplayForModel()

- Edit / Input :

  @HTML.Editor(), @HTML.EditorFor(), @HTML.EditorForModel()

- Display :

  @HTML.DisplayName() : Used to display only the name of model property.

  @HTML.DisplayText() : Used to display only value of model property.

  @HTML.Display : Used to display value with some extra functionality.

  @HTML.DisplayForModel() : Used to display complete model data.

  @HTML.Display.....For() → Strongly Binding version.

- using System.ComponentModel.DataAnnotations; in model class

  → Choose attribute for our model.

Mon Tue Wed Thu Fri Sat Sun

Date:

@HTML.DisplayName("name") // Show the name if this was not written otherwise show this Name string.

[Display(Name = "Please enter your name")]

public string name {get; set;}

In model class "Employee.cs".

- Edit / Input :

@Html.Editor() : Used to create input field using the single model property.

@Html.EditorForModel() : Used to create input fields for entire model.

Mon Tue Wed Thu Fri Sat Sun

Date: 07-05-20

Lecture : 20 07-May-2020

⇒ Custom HTML Helpers :

What if you need something which built in HTML helpers do not provide ?

For example :

@Html.Image()

@Html.AddressFormat() → To display your custom address format etc.

The solution of above problem is Custom HTML helpers means Creating your own HTML helpers as per the need.

↳ How to create :

There are two ways to create custom HTML helpers :

Using static class and static method.

Using Extension method.

- Using: static class, method :

```
public static class CustomHelper
{
```

Mon Tue Wed Thu Fri Sat Sun

Date:

```
public static IHtmlString Label (string src,
                                 String alt)
{
    return new MVCHtmlString (string.Format
    ("<img src = '{0}' alt = '{1}' ></img>",
                                 src, alt));
}
```

- One method represent only one helper.
- Class Type 'static', Method → 'static'.
- Method return type → 'IHTMLString'.
- And return → MvcHtmlString.

- Using extension method :

```
public static class CustomHelper
{
    public static IHTMLString Image (this
    HtmlHelper helper, string src, string alt)
    {
        return new MvcHtmlString (string.
    Format ("<img src = '{0}' alt = '{1}'>
            </img>", src, alt));
    }
}
```

- this HtmlHelper helper

Mon Tue Wed Thu Fri Sat Sun

Date:

⇒ AJAX (Asynchronous JavaScript and XML):

Without refreshing the page, the data display on the page by compiling the server.

- Update a web page without reloading the page.
- Request, Receive data from a server - after the page has loaded.
- Send data to a server - in the background.

↳ AJAX - JS

Call the function loadDoc() on button click.

- STEPS:

1- An event occur in a web page (the page is loaded, a button is clicked)
2- An XMLHttpRequest object is created by Javascript.
3- The XMLHttpRequest object send a request to a web server.
4- The server processes the request.
5- The server send a response back to the web page.

Mon Tue Wed Thu Fri Sat Sun

Date:

6- The response is read by JavaScript.
7- Proper Action (like page update) is performed by Java Script.

↳ AJAX - JQuery :

$ represent JQuery.

↳ AJAX - JSON :

• get JSON( ) function used.

jd → parameter.

∴ Check code for this.

⇒ AJAX in ASP.NET MVC :

? → query string.

Check code for it.

Mon Tue Wed Thu Fri Sat Sun

Date:

Lecture : 21 — 12-May-2020

⇒ Entity Framework :

How to create entity data Model in Entity framework :

Entity data model is a model that describes entities and the relationship between them. (entities = table)

↳ How to create :

- Create a project.
- Install 'Entity Framwork' package.
- Add New Item "ADO.Net Entity Data Model" in models.

* Three approaches to work in Entity framework:

↳ Database first approach : Link the database of previous project in current application.

↳ Model first approach : Software architect create model first by using GUI tools and then create database.

↳ Code first approach : Create table according to model class property.

- Select Database first approach → New Connection → choose 'Microsoft SQL Server' → Give Server

name → Give the database name to connect it.
The file of name 'EDMS' is created

- In model.context file → entity file make that represent conceptual model, The reference of all tables are also include.
- In model.tt the database tables → classes are created

↳ Entity Framework :

- Create simple 'Console Application' project.
- Select the project → Add new Item → Data → ADO.NET Entity Data Model → Empty model.
- Right click → Add Entity → Entity Name 'User' → Set primary key → Add.
- Add New → Scalar property → Name, Roll No.
- Right click → Generate Database from Model → New Connection → Give Server Name '(localdb)\v11.0' → Give Database Name 'User DB' → OK → Next → Entity Framework 6.0 → Finish.
- Click on execute button ▶ → connect.
- Now achieve this table through code.

Check the code

Mon Tue Wed Thu Fri Sat Sun

Date:

Lecture : 22 — 14-May-2020

⇒ ASP.NET MVC with Entity Framework :

- Create MVC project.
- Add Controller and View.
- Add 'SQL Server Database' in App Data.
- Add two tables 'Student', 'University' and foreign key for one to many relationship (Take UID (university ID) as foreign key in student table).
- Add 'ADO.NET Entity Data Model' by clicking the models → 'EF Designer from Database' → Next → Select Tables → Finish.

*' Properties show scalar properties (Table's column)
*' Navigation Properties → Table relation show.

- Add 'using EntityFrameWork.Models' in Controller to give model access to controller.
- Add entity framework class name through which all operation can be done.

```
Database1Entities db = new Database1Entities();
```

- Display student list using strongly type view. In Index.cshtml:

```
@model List<EntityFrameWork.Models.Student>
```

- Now perform create, write, delete and update operation.

Mon Tue Wed Thu Fri Sat Sun

Date:

Lecture : 23 19-May-2020

⇒ Dependency Injection :

It is a software pattern that allows the removal of hard coded dependencies and make it possible to change them, whether at run-time or compile-time.

- Tightly coupled class → Dependency - i.e When ticket is book then email is send.

- System can have hundreds of tightly coupled classes. Because of that accomodating such changes can be time consuming and frustrating.

↳ How to write loosely coupled System :

Classes should always communicate with each other via interfaces.

Interface → No function define so its object is also not created.

Using interface we have decoupled the classes.

Mon Tue Wed Thu Fri Sat Sun

Date:

→ Inversion of Control (IOC) or Dependency Injection :

To use EmailNotification we just have to create object of Booking class.

```
Booking Booking = new Booking (new SMSNotification());
```

↓

In main class ⟶ No changing in Booking class.

DI → Remove code coupling, code complexity and make complex code simple.

↳ STEPS :

- First install by clicking "Manage Nu Get Packages"

  Ninject, Ninject.Web.Common, Ninject.MVC3.

- Add Interface in Models → Interface name must start with 'I'. Give prototype of the function in Interface.
- Add class in Models which provide the implementation of function (interface function).
- Now do some changes in controller :

  Create interface object.

  Write constructor of controller which take this interface object.

Mon Tue Wed Thu Fri Sat Sun

Date:

Write the method which is call against submit button.

- Write a line in NinjectWebCommon.cs (App-Start) → Kernal.Bind<IInterface>().To<InterfaceClass>();

Lecture : 24

21-May-2020

⇒ ASP.NET MVC Scaffolding:

Use previous database.

- Models → Add → New Item → Data → ADO.NET Entity Data Model → Generate from Database → Select table → Finish.
- Add New folder 'Home' in Views.
- Then Add Controller 'HomeController'. If "No model classes are available" shown then select 'rebuilt solution' by right clicking solution.
- Now again add controller 'HomeController'

Mon Tue Wed Thu Fri Sat Sun

Date:

Template :

MVC controller with read / write actions and views using Entity framework.

Model class :

Students (Mvc Application 121.Models)

Data Context class :

Students Entities (MvcApplication 121. Models)

Add this and scaffolding is done.

lecture : 25 28-May-2020

⇒ Semantic Web :

(Semantic Web 3.0)

Every language has its own SYNTAX and SEMANTICS.

Syntax : Study of grammer (is how to say something)

Semantic : Study of meanings (is meaning behind what you say).

Different syntaxes have same semantics.

e:g X += Y

X = X+Y

Mon Tue Wed Thu Fri Sat Sun

Date:

The internet is the huge network of computers that use TCP/IP to communicate with each other.

The web is a system of interlinked documents accessed via the Internet.

Web 1.0

↳ The mostly read-only Web

↳ Web 1.0 is first stage of world wide wibe (WWW).

↳ It is made of web pages connected through hyperlinks

↳ Only static web pages are made up with this

Web 2.0

↳ The widely read-write Web

↳ Web 2.0 describes www websites that emphasize user-generated content.

Web 3.0

↳ The widely write-read Web.

↳ They can learn what we're interested in

↳ They can help us better find what we want.

* Problems of Web 2.0 :

- Web pages are written in HTML.
- HTML describes the structure of the information (Syntax not Semantics).

→ Semantic Web (Web 3.0):

- The Semantic Web is an idea of World Wide Web inventor Tim Berners-Lee that the Web as a whole can be made more intelligent and perhaps even intuitive about how to serve a user's needs.
- It can recognized people, places, events, companies, products and movies etc.
- It can understands the relationship between things.

"The semantic Web is a Web of data, in some way like a global database".