# 

# 4 Pillars of OOP================================================

**Abstraction, Encapsulation, Polymorphism, Inheritance**

**Abstraction - more like ignoring the irreverent or not showing it. private**

**Encapsulation - making sure certain things are hidden from the users e.g DOB**

Both hide private data and expose public properties eg car: gear shift

Create and design correct methods to allow user interaction in the correct manner

**Abstraction (hiding and revealing the bits you want)**

code is hidden behind (encapsulated) but user can interact with it using Public properties

Both encapsulate (hide) key code but also reveal or expose desired 'Properties'

Properties ⇒ {get; set;} ⇒ allows public access

**Car:** engine inner workings are encapsulated (hidden)

steering, clutch, accelerator are public properties accessibly by end user

**Encapsulation**

This is where we hide code which does not need to be revealed to the outside world

**Car Engine :** deep inner workings hidden from general user

**Code :** private keyword for this

4 options with keywords are called 'Access Modifiers'

Access Modifiers

private encapsulates(hides) code from outside

access: code in this class only!

public accessible from any code

protected access: from code in this class and all derived child classes too

internal access: from code inside the compiled .exe or .dll file which is

produced for this project 'assembly'

# Interfaces===================================================

3 Levels of class inheritance

1. Regular class Parent => Child

Polymorphism works here  *(Virtual in parent and override in child) optional*

1. Abstract class Parent => Child

Abstract parent must have a child class to be used.

1. Interfaces

Problem with classes is that we can only inherit from 1 class

*Think natural family : you can only belong to one family*

Interface is a structure that allows us to ‘use’ code from more than one place

*Think garden shed : tools which we ‘implement’ or ‘use’*

Class => Inherit ONE

Interface => implement MANY

Interface IUseThisTool01{

//no fields

//yes properties

Int number01 {get; set;} // Assumed public without the keyword present

//yes methods

Void DoThis(){} // Assumed public

}

INTERFACES ASSUMES EVERYTHING IS A DEFAULT TO BEING PUBLIC

Class MyClass : IUseThisTool01{

Public int number01{get; set;}

Public override void DoThis(){}

}

**Class**: Picture : Class is like the architectural blueprint which an architect creates as a plan for building. It’s not a building but a builder can turn the plan into a real project.

Computing : Class is a template for creating real objects

Class Parent{} Car p = new Parent(); p is a real object INSTANCE created from class template

Class: blueprint or template from which we create real objects using ‘new’ keyword (this is called instantiation)

Class : template

New : instantiate real object

# Docker ===================================================

**DOCKER DESKTOP**14/03/2019

Docker is a command line too eg powershell : Docker or Docker -v

It enables virtualisation on a very small scale

Regualer virualisation

HyperV

ESX-I

VirtualBox

* Run complete operating system eg Linux, Win10 etc, Server2016

Parallers (on apple)

Typical OS is huge : several GB in size

Container Virtualisation

Container Virtualisation

Container Instances, Unique, Shielded from each other kernel (core of operating system, shared Hardware(shared)

Typical container size is a few hundred MB

Lab: find on internet a pre-build ‘container;, download and run it

Connect to the sql server on it using SQL management studio

1. Pull docker image using

*Docker pull kcornwall/sqlnorthwind*

Have you got a valid docker image yet?

*Docker ps -a*

If not let’s first create and run the image

*docker run -d --name sql -p 1433:1433 kcornwall/sqlnorthwind*

If your image running?

*Docker ps*

List running anf non-running images

*Docker ps -a*

Status of docker container

Docker ps

Can you start and stop image

Docker start sql

Docker stop sql

Docker ps -a

Docker ps

# ASP.NET===================================================

15 – 05 – 2019

Most of the world build websites using HTML CSS JavaScript, However there operate inside a ‘browser’ to display the user.

The job of a ‘web server’ is to

* Store fixed pages, images, scripts etc
* Generate dynamically computer generated pages etc.

Client Facebook

IP address 10.11.12.13 🡪 http request 🡪 IP address 67.68.69.70

🡨 http response 🡨

Many sites traditionally use PHP to generate HTML at the server

Java can be used (but not so much now)

Python, Ruby (RubyOnRails) can be used

Google has ‘Go’ and many more

Someone worked out how to use JavaScript on the server also . They called it NODEJS and it has changed development forever.

*Client : HTML CSS JavaScript is running Server: NODE (JavaScript) +HTML CSS*

Now the language doesn’t need to be translated as they have the same languages.

**Microsoft servers run a special language called ASP.NET**

<html>…

<asp:runat=’server’>…button

ASP : Active Server Pages

ASPX = newer XML Version

**Traditional ASP has this format :**

Display Page .aspx xml file

Code behind .cs cs code behing XML

(As seen in WPF)

**Razor Sytnax**

.cshtml with both regular html on the page plus

@{

//put code here

}

Visual Studio we can build several different types of Web Ap

1. Blank
2. ASP Web Form
3. ASP MVC Model View Controller
4. API

First Demo 🡪 Regular ASP (not MVC)

Flat file structure, Each page has a .aspx and .cs file to make it

Second Demo 🡪 ASP MVC site

M Model Data

V View What the user sees

C Controller Sorts out your request, decides which page you want to see, gets the data from the model and sends it to the view for display .

Http:// mysite.com/Controller/Action

http:// mysite.com/api/customers/ all customers

/customers/ALFKI one customer

/Home controller

/Home/About Home controller, About Action which

retrieves the About View

Controller : Deciding which View to use

Home

Index Default

Returns View Page and look for page called Index.cshtml

Pass data from Controller to View

We have seen the default Viewas . ViewData but let’s add to this

Var string01 = “…”

Return View(“Index”,string01);

# ASP Web Core Project=============================28/05/2019

Building an ASP Core website from scratch

We are not going to do a lot of theory but just build the website!

Theory and detail will come later.

Goal

1. ASP Website (not MVC)
2. Flat file Structure
3. Razor Page
4. Use Model (puts data on page)

## Build project (New ASP core web app)

### Folder Structure

Wwwroot

Js Javascript

css

images

FIXED FILES WITH NO SECURITY : COMPLETELY VISIBLE

ALL FILES, FULL PETH LOADED WITH SITE

Pages / Sahred / \_Layout is out file shared acress all pages

Startup.cs

Services

=> Run our database from here!

AddMvc()

AppSettings.Json

Can have database connection string sometimes

Default page is ‘Index’ (can be Default.htm)

Page

Jaspreet.cshtml this is a ‘razpr’ page

Jaspreet.cshtml.cs this is the c# code behind page (real C#)

## Passing data through from our C# code=behind to .cshtml View page

We can create ‘properties’ in the C# page by using the ‘class’ it declares

Class MyPage : PageModel {

Public string Property01{get;set;}

}

## Code 1 – bad table design vs nice

@page

@model lab\_80\_ASP\_Core\_Web.Pages.JaspreetModel

@\*

For more information on enabling MVC for empty projects, visit http://go.microsoft.com/fwlink/?LinkID=397860

\*@

@{

//code

//two ways of sotring trivial strings : viewdata and viewbag

ViewData["Value"] = "here is some data";

ViewBag.Value2 = "And some more data here - ViewBag";

}

<i><b><u>We don't really use Bold Italic underline any more

</u></b></i>

s

<p style="font-weight:bold">@ViewData["Value"]</p>

<p style="font-style:italic">@ViewData["ViewBag.Value2"]</p>

<p>We can also use 'em' for emphasis bold <span><em>@ViewBag.Value</em></span></p>

<p>We can also use 'strong' for emphasis bold <span><strong>@ViewBag.Value</strong></span></p>

<p>We can also use them for emphasis bold <h1>@ViewBag.Value</h1></p>

<p>We can also use them for emphasis bold <div>@ViewBag.Value</div></p>

<p>We can also use them for emphasis bold <p>@ViewBag.Value</p></p>

<!-- p is a block element. always starts at a new line. H1 header also starts a new line -->

<p>Finally to underline we use <span style="text-decoration:underline"> text-decoration : underline</span></p>

<h1>This is my amazing test page </h1>

<p> vyk jbeufvu;ero b;oudfbubu;odfbojfblsfal </p>

<h1> Using Razor</h1>

<h2> The 'at' symbol can be used to write C# code raw on top of HTML</h2>

<h3>@DateTime.Now</h3>

@{

var d = DateTime.Now;

d = d.AddDays(2);

}

<h4>The date in 2 days time is @d.ToShortDateString()</h4>

<ul>

<li>first</li>

<li>second</li>

<li>third</li>

</ul>

<ol>

<li>first</li>

<li>second</li>

<li>third</li>

</ol>

<ul>

@for (int i = 1; i <= 10 ; i++)

{

<li>Your item has number @i </li>

}

</ul>

<h2> This is also Bootstrap</h2>

<style>

</style>

<button class="btn btn-success"> CLICK HERE</button>

<button class="btn btn-danger"> CLICK HERE</button>

<button class="btn btn-default"> CLICK HERE</button>

<button class="btn btn-info"> CLICK HERE</button>

<button class="btn btn-primary "> CLICK HERE</button>

<h2> This is a HMTL table </h2>

<table class="table table-hover table-bordered table-striped">

<tr>

<th>header</th>

<th>goes</th>

<th>here</th>

<th>there</th>

<th>every</th>

<th>where</th>

</tr>

<tr>

<td>also</td>

<td>dvsv</td>

<td>sssd</td>

<td></td>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

<td></td>

<td>fdsf</td>

<td>sdf</td>

<td>fdfds</td>

</tr>

</table>

<table>

<tr>

<th>some</th>

<th>text</th>

<th>here</th>

<th>there</th>

<th>every</th>

<th>where</th>

</tr>

<tr>

<td>lol</td>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

<td></td>

</tr>

</table>

## Code 2 – text with box around it and padding etc

@page

@model lab\_80\_ASP\_Core\_Web.Pages.UsingAModelModel

@\*

For more information on enabling MVC for empty projects, visit http://go.microsoft.com/fwlink/?LinkID=397860

\*@

<hr />

<hr />

<hr />

<!-- use # with id but . with class-->

<style>

div.block {

width: 70%; /\* Note that we can use 50 vm aswell \*/

width: 70%; /\* Note that we can use 50 vm aswell \*/

padding: 3vh 5vw;

background-color: #1e900f;

text-align: center;

font-size: 2.1em; /\*one 'em' means standard size text,

dont use pixels are its hard coded so isnt great for mobiles\*/

margin-left: 13vw;

margin-top:5vh

/\* homework help .css\_flex\_04\_center {

display: flex;

width: 400px;

background-color: lightgrey;

justify-content: center;

}\*/

}

</style>

<div class="block">

@Model.Property01

</div>

<div class="block">

@Model.Property01

</div>

<div class="block">

@Model.Property01

</div>

<div class="block">

@Model.Property01

</div>

<div class="block">

@Model.Property01

</div>

## Code 3 – useful table

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

namespace lab\_80\_ASP\_Core\_Web.Pages

{

public class ListModel : PageModel

{

public List<String> items = new List<string>();

public void OnGet()

{

items.Add("first");

items.Add("second");

items.Add("third");

}

}

}

@page

@model lab\_80\_ASP\_Core\_Web.Pages.ListModel

@\*

For more information on enabling MVC for empty projects, visit http://go.microsoft.com/fwlink/?LinkID=397860

\*@

<h1> Here is a list from a model from the code-behind</h1>

<table class="table table-striped table-hover table-bordered">

<tr>

<th>List of Strings</th>

<th>List of Strings</th>

<th>List of Strings</th>

</tr>

@foreach (var item in Model.items)

{

<tr>

<td>@item</td>

<td>@item</td>

<td>@item</td>

</tr>

}

</table>

## Homework ------------------------------------------------------------------------------------

Can you research and add to your page

1. A FlexBox Div, centered, 100 pixels square
2. Can you import Microsoft.EntityFrameworkCore v2.1.8 into your project
3. Also Microsoft.Entity.FrameworkCore.SqlServer
4. Can you add a folder ‘Models’ to your project
5. In it add two classes
6. Customer
7. Northwind : DbContext
8. (trainer will give you files for 5a and 5b )
9. Using MyProject.Models; at top
10. Using( var db = new Northwind()) {// create a list of customers and display them! }
11. Add code to services method of startup

## ASP Core Entity : Refresh------------------------------------------------------------------------------------

Past1 : Read Northwind

Part 2: Add To Northwind

**Part 1 : Refresh : Read from Northwind**

### Razor pages-----------------------------------------------------------------------------------------------------------------

Great page for razor pages with database CRUD

<https://www.codeproject.com/Articles/1207962/Simple-CRUD-Operation-with-Razor-Pages>

No CRUD ready built in.

Can use injections etc

Make pages yourself

### Add Customers--------------------------------------------------------------------------------------------------------------

1. Add customer object to cs page

//Bind proeprty is required for POSTING of data from form

[BindProperty]

public Customer customer { get; set; }

1. Create form

With a form there are two ways to submit it

1. GET

Submit 🡺 Form and the data get sent COMPLETELY VISIBLE IN THE URL

[www.mysite.com/path?name=Bob&company=sparta](http://www.mysite.com/path?name=Bob&company=sparta)

1. POST

Submit 🡺 Form data is sent as part of the page body in JSON format (key/value pair)

1. Update Post() Method to accept data

# TESTING===================================================

21/05/2019

CI-CD Continuous Integration – Continues Deployment

Every day, the developers write code. They commit it. At the same time they have often built code

TDD – Test driven development

Unit Tests(raw code testing) : create tests in code, makes sure that your code passes these tests

BDD - Behaviour driven development

Cucumber

Selenium

As a … I want to … With outcome… (user stories)

Gherkin Syntax : Write some tests like

As a USER when I click on the ‘login’ button the login form appears

As a USER when I entered my credentials I am authenticated

Run tests

If pass then push code to production => call this a ‘pipeline’

Devops

CICD

PIPELINE

Coders code +push to repo (version control) + test 🡺 PASS

🡺 automate push to production on a regular basis

**QA : Quality Assurance**

Use automated testing to verify the ‘code coverage’ of our tests and ensure high quality of code

Static test 🡺 human being reads your code

Dynamic test 🡺 computer tests your code

**Testing CAN NEVER REMOVE ALL BUGS**

RED writing test first but will fail as no code is present

GREEN write code which passes.

REFACTOR Check our code : Can we make it more efficient / readable / maintainable??

MSTEST Microsoft

NUNIT Industry standard

XUNIT Newer

# LINQ===================================================

21/05/2019

**LINQ Select**

We have used a lot of LINQ LAMBDA but not much of the simpler LINQ plain syntax

***LAB 70 was created here. (IQueryable)***

***LAB 71 was created here. (IEnumerable)***

**IQueryable VS IEnumerable**

*Both use lazy loading ie queries not run until actually needed.*

**IQueryable** Run a query against a remote data base

Only results are returned

Lazy loading : defer until needed

**IEnumerable** Run against a local dataset

Or can specify from a database

WHOLE DATASET IS RETURNED FIST INTO MEMORY

We can enumerate (count) over the output

Lab 70

**- select all customers from these two cities.**

using (var db = new Northwind()) {

var customers =

from customer in db.Customers

where customer.City == "Berlin" || customer.City == "London"

select customer;

//Data actually loaded here

foreach (var customer in customers)

{

Console.WriteLine($"{customer.CustomerID,-15} {customer.ContactName,-20} from {customer.City}" );

}

* **Select one customer via pk**

var oneCustomer = db.Customers.Find("ALFKI"); // only works with primary key

Console.WriteLine($"One Customer is {oneCustomer.ContactName} from {oneCustomer.City}");

* **Orderby**

var allCustomers =

from c in db.Customers

orderby c.City

select c;

foreach (var customer in allCustomers)

{

Console.WriteLine($"{customer.CustomerID,-15} {customer.ContactName,- 30} from {customer.City}");

}

**Doing statistical analysis : GroupBy**

Simple queries have been handled so far but we have not collated and stats eg Sum, Average, Min, Max , Count.

We can go through a data set and create these stats and then group the results to show stats by category

**SMART LINQ QUERIES**

**RANDOM**

String.format($”{field.value}”) has string interpolation in there as well

:n number

:d date

:t time

: percentage

**.NET**

Framework Legacy existing structure in Windows(4GB)

Standard Attempt to unify all versions of .Net going forward

Core New and lightweight version, built for the web Linux, MacOS, Windows.

## LINQ Recap -----------------------------------------------------------------------------------------------

LINQ is Microsoft’s way of talking of SQL database without the hassle of using raw SQL connection strings etc

### Select

var customers =

from c in db.Customers

select c;

### Where

var customers =

from c in db.Customers

where c.City == “ROME”

select c;

### Find

Var customer = db.Customers.Find(“ALFKI”);

### Orderby

var customers =

from C in db.Customers

where C.City == "ROME"

orderby C.City ascending

select C;

### Custom Object

var customers =

from c in db.Customers

select new {

Name = c.ContactName,

City = c.City,

CompanyName //Single values item

}

### Custom Object with class

var customers =

from c in db.Customers

**select new Custom** {

Name = c.ContactName,

City = c.City,

CompanyName //Single values item

}

**Class custom{**

Public string Name { get; set; }

Public string City { get; set; }

Public string CompanyName { get; set; }

}

### GroupBy

Can count stats per item eg Customers per city

Have to ‘GroupBy City’ to make this work

Var customersbycity =

From c in db.Customers

Group c by c.City into cities

Select new {

City = cities.key,

Count = cities.Count()

}

### Join

### Iqueryable

### INumerable

# SQLite===================================================

**Create table**

PS C:\data> sqlite3 test3.db

SQLite version 3.28.0 2019-04-16 19:49:53

Enter ".help" for usage hints.

**View/select table**

sqlite> .database

main: C:\data\test3.db

**Create table.** Integer means the id is incrementing

sqlite> create table animals(ID integer primary key, name text not null);

sqlite> .tables

animals

**Insert Data**

sqlite> insert into animals(name) values ('fluffy'),('bob'),('garfield');

**View all the data in this table**

sqlite> select \* from animals;

1|fluffy

2|bob

3|garfield

sqlite>

# Tech terms===================================================

## Serial computing

Serial means one by one

Serial line : cable or channel where data is sent in individual bit or bytes or packets or ‘buffers’ in a stream

Serial is good for long distance communications

Anything outside your mother board is ‘long’

USB

SATA hard drive

Internet line

## Parallel computing

Inside a mother board the distances are very small but predictable

Create 64 channels and send data 64 bits at a time

= Parallel processing

All computers have today

Also to make computers go faster we add more processors ‘in parallel’

Ie

## Interfaces

I….. at the start (convention)

### IEnumerable

Has a Numeric INDEX eg. Array, List not Queue, Stack

myArray[0], myList[0] not myQueue[0]

Power of array is instant access to any member through the index

RAM : Random Access Memory

### IQueryable

Querying over a database : LINQ query output

### IDisposable

C# is Strongly Typed : All types determined at compile time.

Speed: at run time code is now much faster as much less checking to do on ‘types’

Normal C# realm at RUNTIME : Garbage Collector by itself erase all objects when finished with, on a random schedule set by itself

But!!

Interact with databases, files systems, networks or memory objects : there are outside neat C# framework. TYPES cannot be neatly encapsulated inside c#

Dynamic keyword : treat type as a regular ‘object’ of no fixed type

Eg importing Microsoft Office Libraries

The ‘garbage collector’ does not know when to get rid of memory references to there outside

Implement IDisposable and call the ‘Dispose()’ method manually when done with these objects

### IComparable

Implements the method ‘CompareTo’ where we can create a customer comparison between 2 objects

Apple1.CompareTo(Apple2);

Output 1, 0, -1

1 if apple bigger, -1 if smaller, 0 if same

## Checked / unchecked Lab 75 max length of ints so loops to the minimum (-numbers)

Checked : throws an exception if your number gets too big or small

Unchecked : turns off again (default)

Dynamic : turns off type checking at compile time

Dynamic x = 10;

x = true;

(note : Javascript does this by default )

*This came up in the sparkhire interview …*

**Dynamic language :** loose checking of types: event at runtime the type can possibly change as is the case with JavaScript and also C# with the ‘dynamic’ keyword.

**Statically typed /strongly type language :** types are strictly defined at compile time.

Var keyword: type is still strictly inferred at compile time

**Javascript –** Loose type checking

Typescript – plug in javascript where types are more strictly controlled

Using Strict; is Javascript in ‘strict’ mode

## Static Keyword

Attached to parent class, not instance required.

# .Net Core ===================================================

Useful as it’s cross- platform and now open source

.Net Framework is huge(4.2GB) and Windows only

.NetCore is lightweight and Windows /Linux/Mac

Let’s add these packages

Microsoft.EntityFrameworkCore.SQLServer

Microsoft.EntityFrameworkCore.SQLite

Working with NuGet Console ( case – insensitive )

NuGet : graphic utility

Improve as programmer 🡺 efficient with command line

Use **Get-Package** to see what is installed already

Find-Package EntityFramework

**Install**

Install-package microsoft.entityframeworkcore.sqlite -ProjectName lab\_77\_Entity\_Core

Install-package microsoft.entityframeworkcore.sqlserver -ProjectName lab\_77\_Entity\_Core

Now we have to build our code – but this time from raw!! Quite a lot of work

GitHub : Philanderson888 : c-sharp : Entity\_10\_Core\_Northwind

Products

Categories (related table)

One Product will belong to one category

[Above Certain Columns : add Description or Type of Change of Name]

Relationships

Table relationships expressed with

Class Category{// properties

//end : relationship

Public virtual ICollection<Product> Products {get;set;}

**Configure the database**

Class Northwind : DBContext{

Public DBSet<Category> Categories {get;set;} // create table

}

FluentApi is name given to overridden

# Overnight lab : build your own To-Do Application in WPF==================

**Goals**

Users UserID UserName

Categories CategoryID CategoryName (Home Work Family)

Tasks TaskID TaskName DateCreated DateDone Done CategoryID UserID

(my goal.. add a order by function to orderby date done or and by category for one person)

**Build an app which**

Creates database from a manually run script (either SQL or c# code-first)

Populates database with initial data

Correctly related tables : One task has one user and one category

Minimum : List/ Add/ Remove Tasks

Maximum : FULL CRUD ie List/Add/Remove/Update Tasks+Users+Categories

**Trainer Help : SQL Script**

Use Northwind; // try not to use if can

drop database ToDo

go

create datebase ToDo

use ToDo

go

CREATE TABLE Categories(CategoryID INT NOT NULL IDENTIFY PRIMARY KEY, CategoryName NVARCHAR(50) NOT NULL);

CREATE TABLE Users(UserID INT NOT NULL IDENTIFY PRIMARY KEY, UserName NVARCHAR(50) NOT NULL);

CREATE TABLE Tasks(TaskID INT NOT NULL IDENTIFY PRIMARY KEY, TaskName NVARCHAR(50) NOT NULL, Done Bit Null, );

TaskID TaskName DateCreated DateDone Done CategoryID UserID

# ASYNC===================================================

It’s like a butler.

You tell them what tasks to you need … they do it while you work. – fast – non-stop

Unlike sync – you wait while the task gets done – stops coding process - slow

# Tasks===================================================

Synchronous code runs line by line

A long task causes the [rpfra, tp ‘hang’ which is not good for the user experience

Asynchronous code looks like this in C#

Async void/<task> DoThis(){

await ReadFileAsync(“bigdatabasefile.txt”)

//examples are:

//Http call to remote item

//Network call to item on LAN network

// Database reAad

//Call to memory location

}

1 Main(){

2 DoThis();

3 Console.WriteLine(“hi”);

4 }

Code will run 1,2,3, and then finish at 4.

But is we add Console.ReadLine(); at line 4. It will do this:

Code will run 1,2,3 and then pause at line 4. While paused the async method will return

and we could action it

Critical thing is that code does not STOP!!! Keeps going 1!! Async task returns whenever it wants to.

## Tasks under the hood -------------------------------------------------------------------------------------------

Main() thread 🡺 main process which is ‘service’ (user does not see) or application (user does see)

Service : 24/7 in background : at startup : eg DNS name lookup on internet

Application : foreground : at log on (User logs in) eg Outlook

Main Thread 🡺 want to do a task which takes a long time

C# : Create ‘Task’ object and run it.

Operating system will control this task and its resources ie memory and CPU allocated to it.

Task manager

Main

Sub-threads

Terminology

Service 24/7 in background

Application at log on (User logs in) eg Outlook

Process Running .exe file which takes up Operating Systems resources on computer

CUP, RAM, DISK, Network

Task Separate process but linked to main process

Thread single block of code sent directly to the CPU for execution

(single unit of code being executed)

ONE PROCESS – Many Threads

Idle CPU is idle most of the time!

## Task.Wait WaitAny WaitAll --------------------------lab 79----------------------------

Waits until certain tasks are complete before moving to the next lines of code

# ToDo via ASP using Entity Framework ( EF6 for short) =================

.NET ASP Entity Framework 6

.NET Core ASP CoRE Entity Core

***Difference between Core and Non-Core?***

**Core**

Cross- Platform : Mac, Linus, Winds

Small

Light

New, fresh technologies

Cannot use WPF

Cant link to many heavy native Windows Libraries

**Framework**

.NET is big (4GB) !!!

ASP Web MVC => Import Entity => Scaffold pages => DONE!

## Migrating a Database

Let’s play around with a database

First thing – the database is saved on the local computer so it can be accessed offline and only from within visual studios

Before we do, we want to be sure we have captured old data before we can make changes

Task

1. Save table structure as Drop – Create SQL

Go to SQL Server>into the database> right like on table>Scripts>save it

1. Save data as INSERT SQL

Go on the view data> scripts(button)> save it

Run it again to test

### Updating your code

Save changes to the .edmx file

Delete the relevant controller and re-add it again, it overwrites the views with the new ones.

IF problems:

Delete both .tt files in the .edmx file for the ERD model. And run ‘Add code generation item’.

>data> dbcontext v6 or something

Click on .edmx file and click save

Delete the controller.

Recreate the controller with views

# SOLID Programming Principles =========================================

Theoretical principles of good programming **(S O L I D)**

**S Single Responsibility**

**Summary : Code should be modular and testable but itself (unit tests)**

*Maintenance of your code becomes much easier when induvial classes /modules/methods etc deal only with one function of ‘reason of existence’*

*Update your code => only one single place for this to happen*

*Only affect one module*

**Monolith (opposite of container)**

*Full Application with libraries, databases, attached to an operating system – very big bulky but also if you want to change it, there can be* ***finish***

**Container**

*Small workspace in which we can put an application and all od its dependencies so that we can publish it to the ‘cloud’ and it will not break when moved from development into production*

*App App App*

*Libraries Libraries Libraries*

*===Host : Shared operating systems ‘kernel’====*

**DevOps**

*TDD tests then build code to pass tests (Red tests fail, GREEN tests pass, Refactor efficient). Once test pass, push to productions straight away*

*CI-CD – Continuous Integration Continuous Deployment*

*Team of 50: continually 1) tests 2) pass 3) push to production*

*4) everyone else pull down changes*

*(means incomputable code can be seen asap )*

**Microservices Architecture**

*Large application built from many single decoupled elements*

**Decoupling**

*Code not dependent on other code {valid unit tests but itself}*

**O Open/Closed**

**Summary = > classes open for extension rather than modify root class**

***Open for extension :*** *Derived (child) class : OK to inherit from parent and override behaviour etc*

***Closed for modification :*** *Base (Parent) class : finished (not modified)*

*Illustration : Vistual Studio (Parent)*

*NuGet (plug-ins to modify behaviour)*

**L LISKOV substitution**

*Replace Parent with Child instances : in some situations*

**I Interface segregation (one method only)**

*Keep interfaces with a single responsibility*

*In practise means ONE METHOD PER INTERFACE*

*IDisposable => Dispose()*

*IComparable => CompareTo()*

**D Dependency Inversion**

*Create your code structure with Abstract classes at the top and inherit down with real (concrete) classes below this level*

***Interview knowledge \*\*\*\*\*\*\*\****

S Single Responsibilty

O Open for extension

L Liskov : Parent . child substitution

I Interfaces : one method only

D Depend on abstract classes at top level

# AZURE / AWS ‘DEVOPS’=============================================

ASP Web app => can deploy to Azure

ASP Core web app => smaller, nimble => easier to port to Azure or AWS for hosting

DEVOPS : Build => Tests => Move Database to Cloud => Push Code to Cloud for nothing

Can link to GitHub with automatic push to Azure

# JavaScript (react) ========================================

React is a JavaScript library for placing HTML on the PAGE

DOM

React is a way for js to build html and put it on a page.

# Serialisation, Streaming and Encoding =================================

## Encoding--------------------------------------------------------------------------------------------

**The way we package data. It was package it as a file then we have something called**

start-of-file marker

end-of-file marker

This helps the computer know where the file starts/ finishes

Signature

Group of characters at start of a file which tells any program which cares to look the file type

e.g JPG always starts with FF D8

Which is hexadecimal and in binary this looks like

Decimal binary hexadecimal

1. 0001 1
2. 0010 2
3. 0011 3
4. 0100 4
5. 0101 5
6. 0110 6
7. 0111
8. 1000
9. 1001
10. 1010 a
11. b

FF D8 = 1111 1111 1101 1000

Encryption - Hiding data so it’s obscured

Encoding is the way of laying down data in a structured manner so applications can read and write the data

Two main ways of encoding data:

1. Binary Encoding
2. Character Based Encoding

### Binary Encoding ----------------------------------------------------------------------------------------------------------

Examples of this are .exe or .dll

If we try and open a binary file we find that it’s non-human-readable. In general only applications can ready and write binary files.

Advantages of binary files is that because they are just raw 1’s and 2’s they are extremely fast, at the limit of the computer using them.

Data 🡺 .bin

Audio/Video /Media 🡺 nearly all binary eg .mov

### Character Based Encoding ------------------------------------------------------------------------------------------------

### ASCII--------------------------------------------------------------------------------------------------------------------------

Primitive character set of first computers : simple English letters, numbers and characters from 1 to 127

Includes ‘control characters’ like escape

‘non-printing characters’

CR carriage return

LF line feed

ASCII runs from 0 to 127 which in binary takes 7 bits (2^7 = 128)

### UTF-8--------------------------------------------------------------------------------------------------------------------------

Default for the web today

Simple ASCII plus using extra ‘bit’ in a standardised manner

Now have 8 bits which computers like better

### UTF-16-------------------------------------------------------------------------------------------------------------------------

Default for C#

It’s also default where we need to include languages globally with non-English characters sets

e.g Chinese, Arabic

UNICODE is the name given to the ability of having every language in the world catered for under one

### Summary -------------------------------------------------------------------------------------------------------------------

Encoding: write data eg to a file

Binary (non readable)

Character based encoding

ASCII 7-bit English Characters

UTF8 “”

UTF16 Any Character in world

UNICODE = 16 bit

File Signature : file type

Encoding : lay down a file

ASCII : 7 Bits

UTF-8 : web : 8 bits

UTF 16 : C# : 16bits UNICODE every language

## Streaming ---------------------------------------------------------------------------------------------------------------------

Streaming is sending data from one place to another

Main types of streaming are

1. To network

http://

https://

Another protocol

<ftp://files>

ssh://data to linus

smtp : send email

pop : receive email

imap: receive

Ldap : active directory

Share //server/share UNC path

SMB ‘Samba’ Server Message Block

CIFS Common Internet Filing System

NFS Network Filing System (unix)

1. Filing System / Hard Disk

… File … C# Library

1. Memory of RAM of our computer

Fastest method of storing data because RAM is fastest storage we have

Use raw binary which is fastest encoding possible.

byte[] arrays : just blocks of 1 /0 s

*Buffer – the name commonly given to the size of one byte[1024] array*

### Summary so far ---------------------------------------------------------------------------------------------------------------

Streaming : sending data in blocks of a size determined by our ‘buffer; which is an array of binary bytes[]

**Stream sending data**

Destination types

1. Memory RAM (useful for encryption)
2. Remote Server
   1. http
   2. https
   3. stmp email SEND
   4. pop email GET
   5. imap email GET
   6. ldap
   7. ssh secure shell : port 22 LINUX SERVER
3. File System
4. SQL server maybe

Buffer single ‘chunk’ of data. Data is packaged until it fills one buffer which gets send, then repeat until all data is sent. Last buffer sent anyway even though its not completely filled.

Byte [] array – giving the buffer a size of binarys

**Several ways to serialize an object**

1. To Binary .bin .exe .dll (non-human-readable format)
2. XML readable <>
3. JSON readable {}

### StreamWriter -------------------------------------

Much the same as streamreader

## Serialization----------------------------------------------------------------------------------------------

A serial stream is a single stream of data, in order, sent from A to B.

1. To Disk
2. To RAM
3. To Memory

Computing : Serial wire allows electricity to carry one channel only (USB, SATA)

Class Customer {

Private \_hidden;

Id

Name

Age

IncreaseAge(){}

GetBalance(){}

}

Send data about our customer across internet : how much should we include?

Serialization : Choosing structure which we send

### Serialization And Streaming : Picture -------------------------------------------------------------------------

**Ikea**

Furniture in store Real object (instance of a class)

Flat Pack Serialization (take some or all of members, ready for sending)

Send to house Streaming : text, xml, json

Lorry has max size : Computing ‘buffer’ filled with data

Reassemble it De-Serialize

Constructor : re-assemble the fields

Instruction manual Class : guide for re-assembling

Assembled at Home

### Serializing a class ---------------------------------------------------------------------------------------------------

We can use [Data Annotations] to let .NET know we are planning on serializing a class

[Serializable]

class SendMeAcrossTheInternet

{

[NonSerialized]

private object DontSendMe;

public string SendMe { get; set; }

}

}

### 3 Main Types of serialization-------------------------------------------------------------------------------

When we serialize an object there are 3 main output tupes:

1. Binary (fastest, use to RAM MEMORY, useful in ENCRYPTION when storing ENCRYPTION KEYS in memory)
2. XML standard UTF-8 : data in readable format
3. JSON UTF-8 : data in readable format

### Schematic Example : Just bare bones outline ----------------------------------------------------------------------

Scenario : serialize as BINARY and send to File.bin file

STREAM : FILE STREAM

SERIALIZE : BINARY

Var binaryFormatter – new BinaryFormatter(); // c#

Using (var stream = new FileStream(“File.bin”)){

Var customer = new customer() {

CustomerID = “ALFKI”,

… }

binaryFormatter.Serialize(stream, customer);

};

File.bin contatin binary representation of customer

### Deserialize-------------------------------------------------------------------------------------------------------------------

Using ( var stream = File.OpenRead(“File.bin”)){

Var instance = formatter.Deserialize(stream) as MyClass

}

### Serialize to xml ----------------------------------------------------------------------------------------------------------------

Exactly the same code and syntax; just different runtime methods to use XML instead of binary.

SOAP Simple Object Access Model

Older version of XML designed to send data across the wed : not used so much now for real internet communcations but definitely in use as we see here.

Note that for customer objects we can create 2 methods : one to send and one to receive (constructor) which can customise fields sent/received.

Note : .NET serialization above but also exists another web method for XML serialization.

### Serialize to JSON ----------------------------------------------------------------------------------------------------------------

Built- in .NET JSON Serializer

Also DataContractJSONSerializer : more popular as faster

More recently : 3rd party which is most popular (Newtonsoft JSON.Net)

1. DataContractJSONSerializer

Class is OPT – IN for Serialization

class BlogSite

{

[DataMember]

public string Name { get; set; }

[DataMember]

public string Description { get; set; }

}

1. JSON.NET from Newtonsoft

Must install from Nuget newstonsoft.json

get-package installed

find-package \*newton\* search internet

install-package newtonsoft.json -projectname -… install

# Creating a Database in Azure=========================================

Portal.Azure.com

Trial : can use real email or use a secondary email for a throw-away trial

E5 enterprise Azure trials 🡺 sign up for free 30days with £200 credit

# Web Technologies In General =======================================

## Web words round 1 / 05/06/2019---------------------------------------------

### Angular

{{ javascript right here }}

Render pages/data right on page without rebuilding page

AngularJS 🡺 Original, old

Angular 2/3/4/5/6/7

### AJAX

New XmlHttpRequest() : allows us to

GET

POST

Data to/from a URL (holding API) without reloading the page

### API : GET/POST data via URL

### React :

Virtual DOM : Build some or all of our webpage and then just ‘display’ it instantly without refreshing page

Control over display.

Traditional HTML : page refresh

React : zero ‘lag’ : reload instantly

### State:

HTML and web by design are ‘state-less’

Stateless - Doesn’t track the ‘state’ (data) – ie where you have been/ going/ logged in/ history

So checks everything

Stateful – tracks history (cookies : data sent on the back of every html page request)

HTML 🡺 request 🡺 server

(send cookie also)

<< ====reply ========(include cookie)

4KB max size

----------------------------

Angular came from Google

React came from Facebook

Vue came from one man

----------------------------

### Vue (pronounced at voo)

Vue has ‘taken off’ as supremely important library

Good things from angular /react & removed not – so -good things

### Stack Overflow Developers Survey 2019

ASP Active Server Pages (old)

ASPX XML version (current)

ASP Core Cross platform

ASP.Net

ASP.Net Core

### Web Forms : ASPX : no longer use

### Razor .cshmtl current - .cs code-behind

### MVC Model View Controller

**Model – Classes represent our data**

Customer

Product

Context has instructions to connect to database

Startup.cs 🡺 inject database as a ‘service’

(when injected as a service we don’t have to re-instantiate every time)

Web.config 🡺 path to database string

Appsettings.json 🡺 Link ‘name’ of database to path/connection string

FluentAPI 🡺 create relationship between models

Virtual customer .. shows 1 to many relationship

Entity Code-First : Models locally : run code 🡺 generate database for us

**View – Visual element which the user sees**

**Controller**

1. Looks at the path of the HTML request coming in

<http://mysite.com/45678/Home/Index>

Looks at the path coming in a split it up into components

Be aware of other elements behind the scenes

HTML HEADERS

REQUEST GET/POST

Status 200 OK

404 Error

Content-Type Application/json /html /text /xml

{

“Content-Type”:”application/json”

}

1. Depending on the path – it will run a method (called an action)

IActionResult OnGet() {

return(View);

}

IActionResult OnPost() {

return(View);

}

return(View, Model);

return(“NameOfADifferentView”);

**Summary**

**MVC**

Model : sees the data

View: what the user sees

Controller : take request, pull down from data from Model and sent to View

App.Config WPF

Web.Config ASP

Both are XML files with RUNTIME information : change without re-compiling app

EG Database connection string

System.Web.Security 🡺 Users/Groups/Permissions

### MVC Basics Lab---------------------------------------------------------------------------------

MVC basics : URL

<http://localhpist:12345/Controller/Action/ID>

/Home/Index home page (default)

/ shows same default page

**Lab\_91**

**Task 1:**

Add View eg PhilPage

IN Controller Index Action

Add Viewbag item: get printing on screen

Change the view to PhilPage

Create Model with 2 properties {get; set;}

In controller

Instantiate model

Add some data

Pass data to view Return(View,model)

Display on your view

**Task 2: Homework**

Repeat the above

Create a new page

Add in Entity Models (code first) from Lab 83 : you will need code from :

Northwind.cs

Customer.cs

Startup.cs => the bit where you inject Northwind database as a service

See if you can get Northwind Customers to appear on one of your View Pages

Stateless /stateful

## Web words round 2 / 06/06/2019---------------------------------------------

Stateless pages do not keep record of where its been or going.

Stateful pages – tracks progress

IP which runs internet is by default stateless

Send traffic : two type of internet traffic

1. UDP stateless : fast : eg streaming music
2. TCP stateful : slow : guaranteed data eb banking

HTML default @ stateless

Cookies : add tracking to our website : test data sent with ever page request/ response

Xamarin - used to build Android / IOS apps - mobile operating systems

Android Studio : XML framework using Java for building **native Android apps**

Native IOS / android – Amazing for first time apps

XCode : MAC IDEA for building IOS mobile apps

UWP : Upgrade to WPF : Windows 10 (Desktop and Tablet) application

JSON Web Token : method of authentication used on open web (open course application)

OAuth : Open Authentication : hooks your website up to Facebook, Google etc so users can log in to your site with Google credentials

Hashes mathematically are ONE WAY Encryption only ie Phill22 🡺 hash = gbiu34wkehldxwhgt5b3gn3eo93tb7ocrnphcr

No way of reverse engineering this has back to Phill22

Rainbow table : people store lists of hashes with passwords

**AJAX – Asynchronous JavaScript and XML**

**JavaScript XMLHttpRequest**

**Jquery $. ajax() $.get() $.post()**

**HTML page and inside it, a <div> with data which is fetched dynamically and can change live on page while page just stays put.**

SPA single page application : uses exactly this thought ie it loads the skeleton which stays fixed and content inside can change as you navigate from page to page.

**WebAssembly (the future) : new : compile programming code neatly inside a browser engine : opens up real doors of speed and flexibility. Compiles down to native C++ lightning-fast speeds.**

Blazor pages : use WASM

Clouds

1. Cloud in world : AWS Amazon web service
2. Cloud : Azure
3. Cloud : google cloud

Web worker : javascript working in background

Geolocation : track location

Accelerometer

Low Level Talks to the CPU directly

High Level Human readable code – C#, Java, C++

MSIL Intermediate language

ILDASM A tool to inspect your intermediate language code

Runtime environment which runs your code

Garbage Collector Remove data from HEAP Memory when finished using

MVC

Model Data + context to connect to database

View Visual page

Controller Looks at the incoming URL request and analyses it

Decides which page/data you see. Can fetch data from model into view for display

<http://localhost:1234/Controller/ActionMethod/ID>

/Home/About

Controller is home / Method called is About ‘action method’

IActionResult About(){} //Method with returning view method.

Return View() 🡺 fetches the view inside the Home/folder called About.cshtml

Razor Mix of HTML and C# all on one page @ { ..\\content.. }

Angular Renders JavaScript quickly on page {{…content…}}

React Build Virtual ‘Dom’s

DOM Document Object Model : all web pages have structure based on this

Window

Html

Head

Body

Div

Content

Vue Similar to react, newer

Typescript forces JavaScript to have fixed types like C#

File.ts 🡺 run the code, must compile ‘transpile’ down to javascript

Transpile translate from one form of Javascript to another

Nodejs Chrome : We strip out the working parts and run as an engine directly on a computer

Node node -v

Npm -v

Web server Stores and delivers web pages across the internet to web clients which request data

Web client request web page

Web browser display web page

Web farm Multiple web servers working together (big companies)

Load balance When we get a request for a web page, it is equally shared amongst multiple servers in a web farm

IIS Internet Information Services : Microsoft free web server

Apache Web server : free, open source (most popular) – not linked to any language so can host any website

Language-agnostic Cross platform / cross all languages

Nginx

**IP 1.2.3.4 Unique number given to a machine on the internet**

\\ the same ip but with the others last number \c$

Private IP 192.168.x.y local

Public IP on internet

Gateway links your private network to internet

Port 1-65000 : like a channel

Socket IP : Port 192.168.10.10 : 8080 🡺 unique channel to send data to a computer

Firewall Block IP Traffic using Both Ip and port number mainly

Proxy A Fake IP internally : internal computers link to it : web requests go to the proxy.

Proxy will get data from internet, send back to PC

Reverse proxy PC on web, Authorise by us, pulls data directly from internal database servers

HTTPS encrypted traffic

SSL encrypted data (old)

TLS new version of SSL

CICD Continuous Integration, Continuous Deployment : Push /Pull / Testing all the time

Webpack Bundles right versions of Libraries together

Caniuse.com allows you to build css grid.

# XML ===================================================

XML is used by Microsoft.

XML is completely used to build Microsoft Office!!! .docx is actually a .zip file containing XML

JSON is used by the rest of the world to end data

{

Field : “value”,

Field2: 22.

…

}

XML has a rigid structure

Explore it today

Goal

1. Create xml
2. Read xml

Console.WriteLine("\n\nHelloWorld First Element\n");

var xml01 = new XElement("testElement", 100);

Console.WriteLine(xml01); // <testElement>100<\testElement>

Console.WriteLine("\n\nAdd Sub Element\n");

var xml02 = new XElement("testElement", new XElement("subElement", 200));

Console.WriteLine(xml02);

Console.WriteLine("\n\nAdd Multiple Element\n");

var xml03 = new XElement("testElement",

new XElement("subElement",

new XElement("subElement", 200),

new XElement("subElement", 200),

new XElement("subElement", 200),

new XElement("subElement", 200),

new XElement("subElement", 200)

)

);

Console.WriteLine(xml03);

Console.WriteLine("\n\nAdd multiple Element\n");

var xml04 = new XElement("testElement",

new XElement("subElement",

new XAttribute("width", 200),

new XAttribute("height", 200), new XElement("subElement", new XElement("subElement", 600))),

new XElement("subElement",

new XAttribute("width", 200),

new XAttribute("height", 200), 300),

new XElement("subElement",

new XAttribute("width", 200),

new XAttribute("height", 200), 300)

);

Console.WriteLine(xml04);

# Serialising Northwind Products to XML=============================

Lets move on then to the next stage where we can serialise real objects and also put them in a loop to read many items.

Lets begin with one simple case with just one item.

Northwind – Product and Categories

Let’s use Product and categories

To get the models we must create a Framework project, ad entity ‘code first’, pull down the models and then they can be copied to our core project

Homework

Push/pull to feature branch 1/2/3 times with Dev pull requests 3 times

Done it twice so far

Difficult work : work and reaserch as a group to deserialize products.xml file bacj to products

Tip – remove version with category names in it .. just plain old categoryid

# Azure Ali =============================================

Download Azure Cli

Verify install az

Login az login

Takes you to the browser login

After successful login takes you to

<https://docs.microsoft.com/en-us/cli/azure/?view=azure-cli-latest>

PS C:\Users\JRai> az login

Note, we have launched a browser for you to login. For old experience with device code, use "az login --use-device-code"

You have logged in. Now let us find all the subscriptions to which you have access...

[

{

"cloudName": "AzureCloud",

"id": "e05f6a66-3152-42d0-8e3a-902814b4b9a8",

"isDefault": true,

"name": "Free Trial",

"state": "Enabled",

"tenantId": "ff15c67c-2870-4e9f-adc1-7d61d855b667",

"user": {

"name": "JRai@spartaglobal.com",

"type": "user"

}

}

]

PS C:\Users\JRai>

## Create ‘bucket’ to hold resources ‘Resource Group’ ---------------------------------------------------

az group create --name ResourceGroup01 --location “West Europe”

{

"id": "/subscriptions/e05f6a66-3152-42d0-8e3a-902814b4b9a8/resourceGroups/ResourceGroup01",

"location": "westeurope",

"managedBy": null,

"name": "ResourceGroup01",

"properties": {

"provisioningState": "Succeeded"

},

"tags": null,

"type": null

}

## Create Virtual Machine ---------------------------------------------------------------

Az vm create -n LinuxVirtualMachine01 -g ResourceGroup01 --image UbuntuLTS

Az vm create --admin-username jaspreet -n jkrai -g ResourceGroup01 --image UbuntuLTS

{

"fqdns": "",

"id": "/subscriptions/e05f6a66-3152-42d0-8e3a-902814b4b9a8/resourceGroups/ResourceGroup01/providers/Microsoft.Compute/virtualMachines/jkrai",

"location": "westeurope",

"macAddress": "00-0D-3A-27-8E-AA",

"powerState": "VM running",

"privateIpAddress": "10.0.0.4",

"publicIpAddress": "104.40.192.92",

"resourceGroup": "ResourceGroup01",

"zones": ""

}

## Create SQL server ------------------------------------------------------------------------------------

az sql server create -n JaspreetKaurRaiSqlServer --resource-group ResourceGroup01 --location “West Europe” --admin-user jkrai --admin-password PasswoPassword1rd1

{

"administratorLogin": "jkrai",

"administratorLoginPassword": null,

"fullyQualifiedDomainName": "jaspreetkaurraisqlserver.database.windows.net",

"id": "/subscriptions/e05f6a66-3152-42d0-8e3a-902814b4b9a8/resourceGroups/ResourceGroup01/providers/Microsoft.Sql/servers/jaspreetkaurraisqlserver",

"identity": null,

"kind": "v12.0",

"location": "westeurope",

"name": "jaspreetkaurraisqlserver",

"resourceGroup": "ResourceGroup01",

"state": "Ready",

"tags": null,

"type": "Microsoft.Sql/servers",

"version": "12.0"

}

## List server -----------------------------------------------------------------------------------------------

Az sql server list | out-gridview

PS C:\Users\JRai> "name": "jaspreetkaurraisqlserver",

>> "resourceGroup": "ResourceGroup01",

>> "state": "Ready",

>> "tags": null,

>> "type": "Microsoft.Sql/servers",

>> "version": "12.0"

>> }

>> List server -----------------------------------------------------------------------------------------------

>> Az sql server list | out-gridview

>>

# NOTEPAD ===================================================

Class : blueprint to create objects

Instance – create object with new keyword

Constructor : special method inside class which is called with the new keyword

Overloading: multiple methods with the same name. but only one with the same parameter as the one being called will be used

Access modifiers – affects ‘visibility’ - public, private, protected, internal (assembly – exe/ dll)

**Abstract – forces a certain structure in your code – declares classes and methods which have to be implemented.**

Abstract Void DoThis()

Child: Override Void DoThis(){} // mandatory

**Concrete** (opposite of abstract)

**INTERFACE : Fully public, fully abstract, can implement many**

**Inheritance : one parent**

**Interface : tool : use many tools**

**AGILE –**

**SCRUM -**

Product owner – with client and manage backlog

**SPRINT**

* **Backlog (list of features to build – user stories)**

**4 meetings**

* Sprint plan – start
* Daily Sprint
* Sprint review - show customer our code
* Sprint retrospective – good/bad internally

## Random words ---------------------------------------------------------------------------------------------------------

class Parent{}

class Child : Parent {}

Members inherited: Properties, Methods, Methods and public/protected/internal fields

Access Modifiers : public private internal protected

internal - in.exe 'assembly'

protected - parent to child

abstract class - no instanite

1 or more abstract method - no code implementation body. {}

- can have normal and abstract methods

interfaces

-public and fully abstract. no fields.

-inherits from multiple classes.

IUSETOOLSET {

Tool01();

Class Realclass : IUSETOOLSET{

TOOL01(){} //class has to implement the method in the interface

}

IDisposable enfore Dispose()

IComparable COmpareTo()

IEnumerable GetNumerator() ..count over array

SOLID

I : Interfaces single responsibilty

struct

normal class

class myClass {

public int property {get;set;} //Property

public myClass(){} // Constructer

DoThis (){}// Method

}

stored on the HEAP memory

struct class

struct myClass {

public int property; //fields only and all are public

public myClass(){} // Constructer default is present even when we make one.

DoThis (){}// Method

}

stored on the STACK memory

stack : fast, small eg int bool

Heap : slower, larger eg array collection string char

value(or primitive) : int bool

reference type : string char

interface - abstract fully -

static public int Age;

class Hello {

}

Loop allows us to repeat the same bit of code over and voer again

for - fixed number of loops

foreach - depends on the collection - every item in collection. such as array list etc

while - if it meets the condition then will execute the loop

do while - does the action and then checks to see if it is accepted

( its great when u want atleast one more loop)

## 05/06/2019 ----------------------------------------------------------------------------------------------------------------

**Stream sending data**

Destination types

1. Memory RAM (useful for encryption)
2. Remote Server
   1. http
   2. https
   3. stmp email SEND
   4. pop email GET
   5. imap email GET
   6. ldap
   7. ssh secure shell : port 22 LINUX SERVER
3. File System
4. SQL server maybe

Buffer single ‘chunk’ of data. Data is packaged until it fills one buffer which gets send, then repeat until all data is sent. Last buffer sent anyway even though its not completely filled.

Byte [] array – giving the buffer a size of binarys

**Several ways to serialize an object**

1. To Binary .bin .exe .dll (non-human-readable format)
2. XML readable <>
3. JSON readable {}

API Application programming interface

API used for delivering data in JSON/XML format for use by an application

API is used for website – used for displaying web page using HTML

Render : HTML : as web page

HTML : Hypertext markup language

CSS

Javascript

**IDE Integrated Development Environment**

IDE : Build Code

Intellisense – pre-shows you suggestions when coding

Structure for building ‘solution’ where join projects together

Add testing

Add running your app

Add 3rd party libraries eg nuget

Debugging

GitHub : deployment

Text Editor 🡺 plain text building code

# Random===========================================================

## RESTful API-------------------------------------------------------------------------------------------------------------------

REST Representational State Transfer : allows you to send data over http

Rest method

Check out postman

GET **read**

Get /customers all

Get/customers/ALFKI 1

POST **new**

Post/customers create new (as no assigned id)

Post/customers/ALFKI edit 1

PUT Put/ customers/ALFKI edit 1

DELETE delete/customers/ALFKI edit 1

HTTP REST API

Data is sent to client in JSON format

{

“key”: “value”,

“key2”: “value2”

}

Regular website : data is sent from server to client in HTTP format

<html>…website

Microsoft website : data is sent from ASP/ ASPX server to client as HTTP

REST API : data is sent from server to client as text data in JSON {…data …} format

EG Postcode.io API

Northwind ASP A

Note : in order to deliver multiple records we use an array

[

{ “key1” : “value1” } , first

{ “key2” : “value2” } second

]

Multiple records e.g

{ “key2” : “value2”, “name”: “bob ” , “age” : 15 }

## AJAX---------------------------------------------------------------------------------------------------------------------------

Asynchronous Javascript And XML

AJAX is very useful to put dynamic content onto a static wevoafe

Page does not refresh

## Tuples---------------------------------------------------------------------------------------------------------------------------

When we return a method we can only return one type as the output

C#7 has created an easy way to return an anonymous type of whatever structure we want. With multiple fields inside it

### Truly Random=----------------=---------------=---------------=---------------=-------------=------------------=

Null coesling

# Things to remember/practise=====================================

## Remember -----------------------------------------------------------------------------------------------------------

Instantiating Var c = new Class() *you create a constructor*

abstract void DoThis(); *It’s doesn’t use the code implementation {}*

## Practise-----------------------------------------------------------------------------------------------------------

Dynamic

Ternary operator (condition)? Action 1 : action 2

List<Customer>

Using (var db = new db())

Arrays + collections

Loops - for foreach while do while

While – int counter = 0;

While (counter <100) {

Cw (counter++)

Do While - do { //code } while (counter <10) .// does the code atleast once

fizzbuzz

# Interview Prep ============================================

My work

* How I did it
* (c# wpf--- entity via sql server )

**Weekend things to do**

* **Loops**
* **If statement ( compound)**
* **exception handling**
* **string manipulation**
* **libraries**
* **try catch finally with a test to see if the checked function throws an integeter overflow exception**
* **length of a string**
* **substring**

Engaging - be sure to engage positively the interviewers. If possible 'meet them' earlier on LinkedIn so you have an idea of what to expect of who the people are before you meet them.

Be sure to make eye contact when you first meet them, shake hands positively and try to relate to them as a human being rather than just someone who is interviewing you.

Expect the technical interview to be tough so check out all the following structures and know that you know them : Strings, Char, Byte, Boolean, Array, List, Stack, Queue, Struct, Enum, OOP, Class, Method, Fields, Properties, Constructor, Instance, Static, Abstract, Polymorphism, virtual, override, abstract and override, Interface - just be aware of, not in detail.

Loops - be prepared to do one on the wall. If they asked you to do 'Fizz-Buzz' on the wall would you manage? Try it in code and then on your wall.

Remember there are four loops : for, do, do.while and foreach over an array.. For loop is over fixed number. While and do.while are over a condition eg loop while records exist to be read from a database.

SQL - know the basics.

Entity - just be aware of the basics, what Entity is (allows you to talk to a database easily, it's a Framework) and that it provides scaffolding ie builds your pages for you. Entity from Database builds code from tables. Entity from Code builds the tables from code.

Why are you at Sparta etc - perhaps brush up on all these questions via a video on your phone. I know it sounds and feels silly but it can perhaps just get you over those hurdles of explaining yourself.

* How did you end up at Sparta?
* Why did you choose a career in tech?
* Can you show your projects from Sparta or discuss work you’ve done on site?
  + What did you achieve?
  + Why did you do it that way?
  + What would you change?
  + What did you find difficult?
* How can you improve on the code?
* How would you test the code?
* What are the most difficult coding challenges you have overcome?
* What quality will you be bringing to the company?

This team will be working together on a new project to try and deliver work in a more Agile and DevOps fashion within Pantheon. All roles will be in Central London, Moorgate. Four other Spartans (3 x C# Dev, 1 BA) started on Monday 8th April and you would be joining this team.

Break it down and talk through what it does one step at a time. Explain it line by line. Ask sensible questions about what’s happening within the code if you’re unsure. They’re looking for you to engage with it and have a credible attempt at understanding it. Work with them to help unravel what it does. THIS IS FEEDBACK FROM A SUCCESSFUL SPARTAN ON SITE AT PANTHEON!

Similarly, they might ask you to write a piece of code on the board. Follow the same approach as above.

**DB Exercise**

They may explain their business model to you and ask if you were to make a relational database from this information, how would you structure it. So use related tables with many to many and one to many relationships. Throughout this asked questions to ensure that you completely understand what they want. They're open to questions as long as you're not clueless and you seem eager to learn what you don't know**.**

**KNOW YOUR SQL AND BE PREPARED TO ASK QUESTIONS!** They want a candidate who will work to understand the core issue of something rather than getting code to work without understanding why.

**Questions for them!!!**

An EXTREMELY important part of any interview is asking the interviewer questions. Please prepare at least **5**, well thought and appropriate questions to ask. Some examples could be:

* What current projects are you working on?
* How are teams structured?
* What’s the company culture?
* Why do YOU enjoy working at Pantheon?
* What techs would I be exposed to?

**Pantheon interview**

4 people but 2 ask tech questions

How would you set up the testing framework

How you would fit yourself in a dev team

How you would set up the next project

25mins long some have up to an hour

How would you make sure your code is testable

Equity -> shares ie buy/sell shares for clients

Shares : Public (traded on Stock Exchange)

: Private – not listed