→ MID - II

Date 20
M T W T F S S

JSON

JavaScript Object Notation [text format!]

It is also a way to serialize your data!

Recall → Partial Postbacks.

We use json to write the body of our request when we use AJAX.

send data to & from a webpage / server in background!

AJAX → Asynchronous Javascript & XML.

→ A very lightweight way of storing & transporting data.

server → webpage.

~ Quite easy to understand. → self describing.	Easy for machines to parse & generate too

Rules for JSON.

data: name-value pairs

comma separated

objects : { }

arrays : [ ]

→ "name" : "value"

→ object { "n" : "v" }

array ["name"]

- an object can have arrays.
- an array can have objects

{ "n": [{"a": "b"}] } ✓

{ "n": [{"a": "b"}] } ✗

You can even write your own code for JSON serialization & deserialization. It can be that simple!

→ Distributed Systems

Date 20
M T W T F S S

↓

Distributed Computing.

→ Multiple Components on different machines. communicating & coordinating together.

→ Appear as a single system to the user.

What makes up a distributed system?

→ Computers / servers / VMs

→ anything that can connect to the internet

→ anything that can communicate by messages.

Scaling ⟷ Horizontally / Vertically.

↓ | ↓

adding nodes | upgrading nodes.

Client - Server Architecture.

Client ⟷ Server.

↓ ↓

Contacts Server | listens for connections.

displays info | responds to users.

Three - Tier. → Kind of like MVC

Web apps!

view on your device + logic on a server + database for storing info.

Date ____ 20____
M T W T F S S

In Three-tier architecture:

- what we see / actual servers / actual DBs.
  - presentation
  - processing.
  - data management

How is it different from:

- View
- Controller
- Model

} all are codes within folders

Web Service

↳ in the middleware

↓ remember distributed systems.

single system → then → multiple systems

↳ front end | processing / business logic |.

→ C#, Java, etc were made to handle programs on distributed systems.

→ So while C# / Java is platform independent, they still cannot communicate with each other through normal means

Example:

C# Object Creation →

- You create an object → constructor is called.
  - ↳ Now in distributed systems, you have multiple systems calling multiple constructors.

→ Normally, destructor for the object is called by the system that created the object.

Server A : created object ⟷ (using from) Server B.

↳ must call destructor.

Date 20
M T W T F S S

Limitations:

↳ Granularity : when you use resources you must remember to release them otherwise, resources may end up occupied & end up crashing your server

Web Services → C# & Java development, while individually platform independant, would have trouble communicating with each other.

→ That's where web services come in.

↳ a system providing services ↲

platform & language independent

→ You can call it from any platform.

→ It does some processing & returns the data required or specified.

↔ You also get rid of the problem with resource consumption. [like object constructor/ destructor]

↔ On WEB: so it's platform independent (windows/ Linux)

Port: 80 ↳ for web { web Protocols: HTTP/ HTTPS

Recall → windows Services had no UI

↳ Similarly, web services have no UI!

↳ Just services! ↝ processing.

↝ authentication, scrapping, etc.

Date 20
M T W T F S S

How are we passing data?

↝ In the client-server code, we were just passing it into a network stream through a Stream writer.

→ But for multiple systems we have to go towards something more generalized & understandable by all machines / languages.

→ Recall : XML & JSON

↳ structured formats for passing data

↳ had tags

<student> </student>

<roll no> </roll no>

↳ order doesn't matter, but data is getting transferred properly → same.

objects → name/value pairs.

XML ↔ information.

SOAP → is a protocol.

Tells what things are available in XML

ASMX → Is based on XML.

↳ XML based web service.

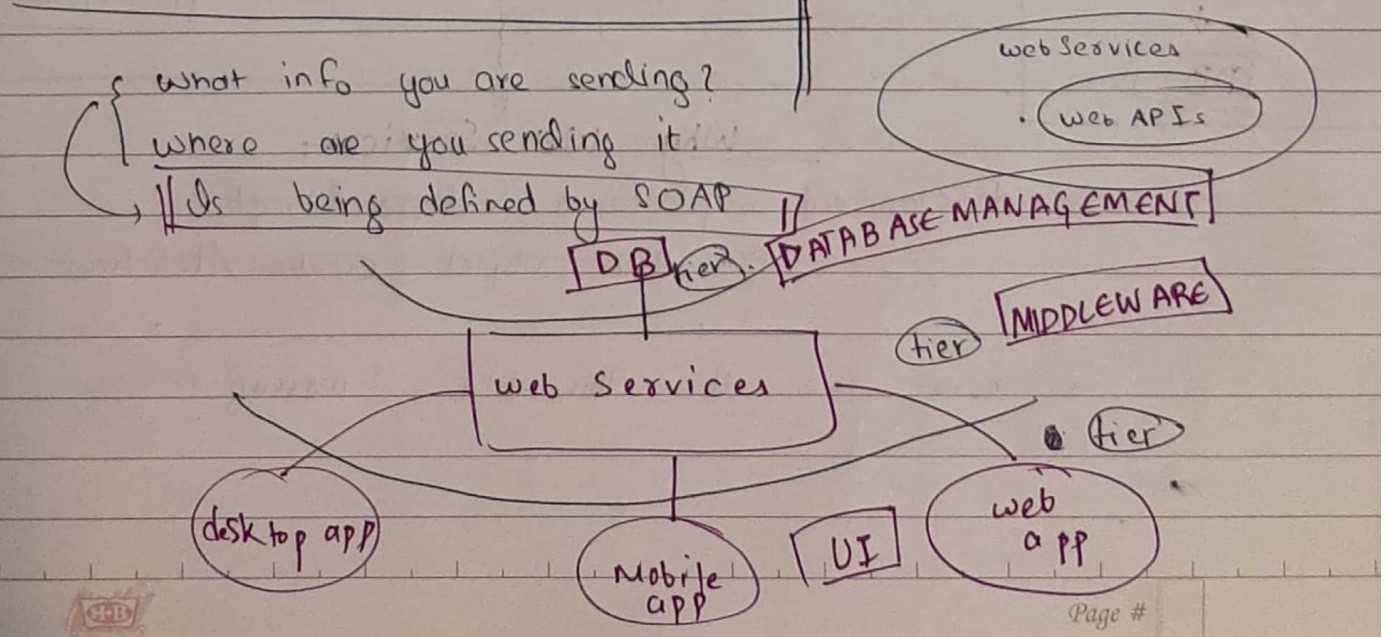
JSON will also be passed. It came after XML web services.

They end up being called WEB APIs.

↳ what info you are sending?

↳ where are you sending it.

↳ Is being defined by SOAP



Date 20
M T W T F S S

API example: Login with Google (for example)
Google will provide an API for it, which you integrate with something!

Desktop App ⟷ Web Services
Communicate via XML

↳ has methods/objects.
↳ Is being serialized in the background.
↳ Deserialized by webservices.

.NET will be handling the conversion.

Project Architecture.
UI
↓
Web Service → APIs
↓
DB

Web Services v/s API
~ Protocols & Transport Mechanism
~ superset of APIs

FB login. (web)
Google Maps.
or windows application.
→ let's say: pop up box

Why Web Services:
~ development platform independence!
no more object creation/deletion

~ now we call this data, "message"

Date _____ 20___
M T W T F S S

→ Object State & Scalability

Servers:

- **Stateless:**
- **Stateful:** maintains values { for example a global count variable. for security reasons → sessions in web

[Usually preferred for Banking etc] 90%

Session is created on login.

→ no need to login again ← State is being maintained on the stateful server.

would make it slow eventually.

- Client would be maintaining that info.
- Each request is a new request.
- Decreasing load on server

facebook - login

↳ cookies, etc for stateless.

ASMX is made on top of ASP.NET {{ stateless by default.

You can introduce stateful services to make it stateful.

|| We are moving to RESTful services, which are stateless ||

Interoperability

↳ language / platform independence.

→ At first CORBA was being used. [not proprietary]

but had framework limitations.

→ XML can be used for all languages! Java / C# , etc

Date 20
M T W T F S S

Industry Standard Messaging

Port 80 is always available.

~~HTTP~~ / ~~HTTPS~~

So now we are exchanging messages b/w different platforms & languages.

You expose the web services on port 80
everyone can communicate.

Benefits

⇒ Everyone can work on their bits of functionality.
⇒ Everyone can call methods
⇒ Business knowledge shared by all members working on a system.
⇒ one DB in the background.

An application just calls the methods we make!

Being handled by W3C is standardized.

SOAP → Simple Object Access Protocol.

Protocol for information exchange

→ Defines ~~not~~ how to send messages.

WSDL → Web Services Description Language

HTTP , XML , SOAP & WSDL

↓ defines where a ~~web~~ web service is hosted, what methods it has

does not define the method code.

Date 20
M T W T F S S

Service Oriented Architecture

You make your software on the basis of "services" that you need to provide.
This way you can cater real-world processes better.

Windows Services / Web Services are the prime example of Service Oriented Architecture.

↳ are exactly that! functions for something you need to do.

Services.

- No UI
- Can be deployed anywhere in the world
- UI for a software can be on one PC & service can be on another PC

ASP.NET Web Services

Like you need WAMP/XAMPP for Php websites, you need IIS / IIS Express for ASP.net web

↳ Development Only

[WebMethod]
↳ exposes methods on the API

wsdl

Viewing ~~WDSL~~ WSDL : In your URL webserviceName.asmx?

The WSDL is shared normally by default
But you can hide it for security reasons
Again, by [WebMethod] → May be used to expose.
↳ Not exposed if not mentioned.

A method which is not exposed, can be called internally.

Date 20
M T W T F S S

In WSDL The methods without the [WebMethod] tag do not show up.

"SOAP DETAILS"

→ A way for running applications to communicate

[Soap Envelope] → Every Message Must be in a Soap Envelope

- Header → sensitive data
- Body → actual body
- Fault → errors

SOAP is XML based.

- → encoded using XML
- → Must have a SOAP envelope namespace
- → " " SOAP encoding "
- → Should not have a DTD reference (schema ref)
- → Should not have XML Processing Instructions

SOAP Header is optional, not compulsory.

m: Trans → transaction.

mustUnderstand tag →

Example

Like in banks ~ to protect sensitive data

By default, mustUnderstand is "0"

Server 1: America

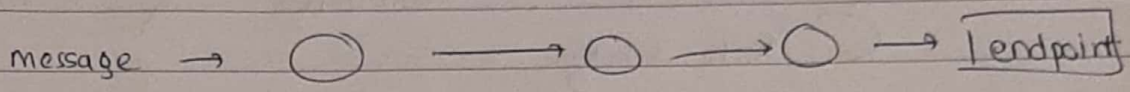
multiple Servers / routers / ISPs.

Server 2: Pakistan

Should all understand it or not?

If mustUnderstand =0 all understand it, it will be passed, otherwise discarded.

"Actor"



all parts of the soap message may not be intended for the final endpoint. Might be for some of the points in the path.

'Actor' is used to address a specific endpoint.

In .aswx , it is default

SOAP Body → Intended for the ultimate endpoint.

for Errors:

Error Codes | Short description |.

In SOAP body:

<fault Code>
<fault String>
<detail> → stacktrace.
<actor> → where it was called.

SOAP runs on HTTP & SMTP normally.
usually blocked.

Does not run with TCP

Date 20
M T W T F S S

OBJECTS, VARIABLES & DYNAMIC.

↳ Root datatype.

{

Recall → Value & Reference Types.

Introduced in (1.0) (.NET Framework).

→ Determined at run time.

- Can hold anything.
- You need to type cast when you're using it.

Variable

↳

Introduced in (3.0).

- Can hold anything but
- Data type is predetermined at run-time.

Dynamic

Introduced in framework (4.0).

- Can hold anything.
- No need to initialize.
- Types determined at run-time.

Date 20
M T W T F S S

• To discuss:

→ what information does an object lose when being cast to an object -?

Values = 1
"A"

DateTime.Now
Class ();
new ~~Employee();~~
→ normal initialization.

```
int i = 10;
i. methods;
obj o = i;
o. methods;
var v = i;
v. methods;
dynamic d = i;
d. ;
```

(no methods)

You'll get errors at compile time here

You'll get errors at run time here!

Compiler doesn't have any information. You can call anything from it.

Ref v/s Out

Passing a parameter by reference.

→ must be initialized before you pass it

→ Initialized before returning to calling method.

→ when you want to return multiple types

{} newer versions have tuples

pass an int

```
static void methodRef (ref int num)
```

assigned / unassigned

```
" " " (out int num).
```

must be changed.

Mid II x

Date 20
M T W T F S S

«WCF» Windows Communication Foundation

This is another type of a WEB service. | came after [asmx]

- → has been around for a while
- → is used in a very secure environment.
- → used often in the market
- → not preferred for light weight things.

Object-Oriented | Component-Based (distributed architeture) | Service-Oriented.

- .NET
  - ↳ asmx
    - ↳ WCF.

Example: No everything is web-based.
Some high security environments block different ports.
- Like a bank wouldn't allow you to communicate to other things out of security concern

Different Models —? ASMX & Remoting → Legacy systems. { Both .NET but very different

So they had to make a way [a service] that let the two models communicate together.

(WCF)

Example Scenario # 1.
- Asmx service interacting with Java app
- communicating via HTTP > XML & SOAP.
  ["passing messages"]

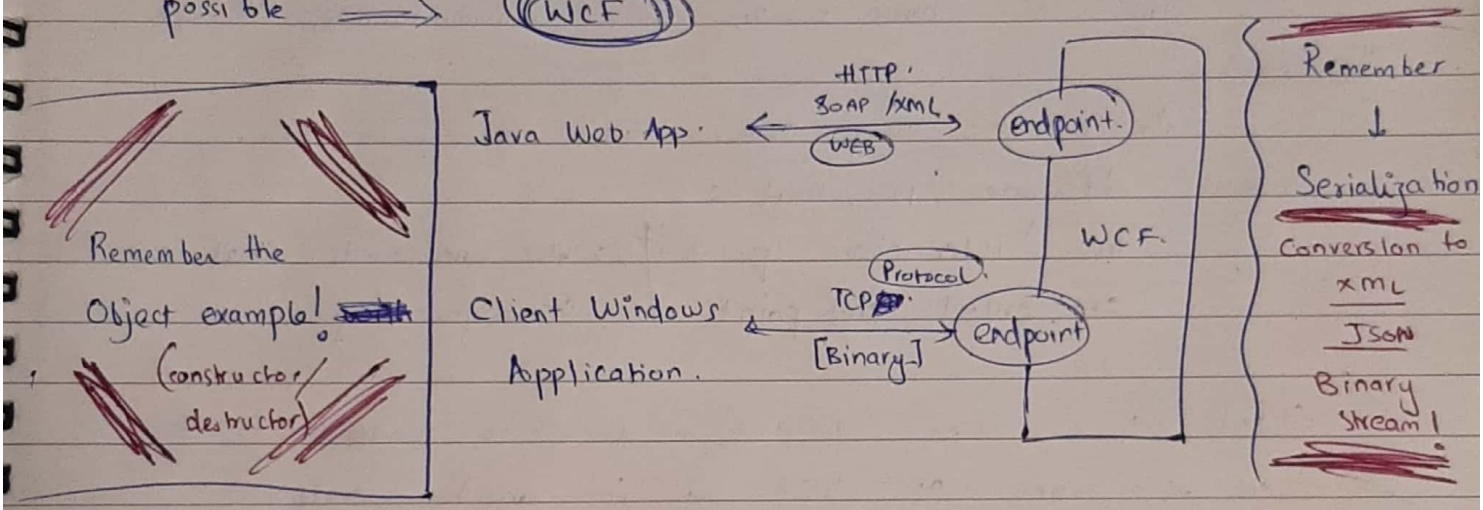
- DotNet Remoting Service. (Objects) - Legacy system.
~ (TCP) ~ communication

Date 20
M T W T F S S

So you end up with two people that will be doing 2 different things.

- one for web servic [asmx].
- one for remoting Service [object-based]

Benefits of Remoting Seovice was that it could communicate over TCP which (Asmx) couldn't do.
So they made something that could help in making both things possible ⟹ WCF



WCF → Merged different protocols with its introduction.
↳ in a secure

WCF → Can replace asmx.
• it can do whatever asmx could do! And More!
↳ remoting.

WSE → Web Service Enhancement → session management
↳ was implemented by asmx later
↳ but also was included in WCF

IPC → MSMQ
↳ Inter Process Communication
Microsoft ~~Management~~ Messaging Queue.
→ Communication between different processes on a system.
[WCF can do that too].

Date 20
M T W T F S S

COM [legacy systems] also supported.

- asmx is by far the easiest to make, implementation wise.
- WCF, secure, powerful → lots of customizations, you can handle multiple protocols as well.

→ WCF Principles

like in asmx → we were communicating through "messages".

It is the same here. You're passing messages.

- But difference is that:
  the "interaction points" are called end points → the two points that are communicating.

Client ↔ runs on HTTP → is one endpoint.
Client ↔ runs on TCP → is another endpoint
client ↔ any other protocol → " "

|| Service can have multiple endpoints ||.

Endpoint has 3 things:

Address (where) || Binding (How) || Contract (what)

Also known as the: ABC of WCF (aka endpoints)

Address → In the Transport Layer → where to send.

Binding → Protocols & Encoding mechanisms

Contract → What methods to show/are visible to user.

⇒ Making a WCF Service

→ You can create it directly from the "New Project" options.

→ When you run it :

- May run on browser

OR

- Might open a built-in testing tool : (Desktop Application) WCF Test Client

Visual Studio
Cmd prompt → wcf test client

Codes

Ex#1 : Service Contract

Ex#2 : Data Contract.

Duplex Service.

[Report Service]

↳ IReportService [OneWay = true]
ProcessReport();

Client → IReportService.

In Duplex → [You pass your session information]

It subscribes the client to the server. So server can send the stuff to the client

Address, Binding , Contract.

↓ ↓

Port + IP Protocol (HTTP / TCP).

Service Contracts ↓ [are interfaces] [OperationContract]

Data Contract — Custom Data Types Not in SOAP [DataMember]

Message Contract — SOAP Modifications

*
- Service Contracts are necessary.
- Data/Message Contracts are not necessary/compulsory

Address.

→ Party is hosted on some URL
  ↳ can be local if hosted locally.
  - So other services can communicate with it.

asmx → was xml Based. // extension was: .asmx
wcf → also was xml Based // extension → .svc
  ↳ but then some changes occurred.

Contracts.

→ which endpoint is communicating.
→ Remember [WebMethod].
→ Also defines function behavior.

↙ ↓ ↘

Service Contract | Data Contract | Message Contract

[WebMethod] ↓ WSDL ↓

SOAP
↳ header / body, etc.
you can control via wcf

[Service Contract] • Equivalent to WebMethod
Interface

↓

[Operation Contract] → on top of methods.
→ if not present, will not be included on the WSDL.

Date 20
M T W T F S S

SOAP defines all existing data types.

→ So → for user defined types, you have to send information about them.

↓

DATA Contracts !!

You send user defined classes with them.

[Data Contract].

Like in Serialization

[Data Member] → the ones you want to share.

↳ Do not add on ones you do not want to share

[Data Member]

Will share the class with the consumer.

Message Contracts.

→ to customize SOAP headers / body, etc.

→ Could not be done in asmx.

→ Not done very often.

Differences b/w asmx & WCF

↳ Request & Reply [Client] [server] | Default Behavior

- Request & Reply (2-way)
- One Way.
- Duplex

↳ server initiates the message--!

(Notifications on different apps).

Date 20
M T W T F S S

New → WCF → [No UI]

"Proxy Design Pattern"

When we were studying Distributed Systems - ?

- 3 tier architecture.
  - Presentation
  - (Web Services were here) Middleware
  - Database layer.

You only have to rewrite the service.

~ New methods needed? Write / Update Service
~ But do not impact existing systems.

→ You're trying to manage user rights on a service level.

Func 1 ( ) { } available

Func 2 ( ). { } not available

→ How to do this?

~> Make a proxy class in between!
Client will alway only be able to access the proxy class..

Real Subject → Actual Class.

interface → will have all methods.

Proxy → You make an object of this.

Date 20
M T W T F S S

What's the advantage - ?

- If you have too many objects, it may cause slowness.
- Always invisible to client [abstraction]

Remote Proxies :

↠ Create a proxy for your object that is located somewhere else in the world.

→ Marshalling : memory representation
↓
format suitable for transmission / storage.

Virtual Proxies:

↠ Provide some default / dummy data if the real object is busy / taking some time to do something.

Protection Proxies:

Managing types of access :

let's say:

"Studen" "Admin" & "Teacher"

→ Desktop
→ web / Mobile.

---

(1)

In the Code: You have a client class that calls the Request method for Subject.

(2)

Calculator Example <-> Interface IMath.

Date ____ 20____
M T W T F S S

Collections

- Read - Only Collections.

→ Do not allow users to ~~edit~~ edit the collection.

- Synchronized Collections:

↳ to keep collections thread safe.

You use wrapper classes to make them thread safe.

Distributed Objects.

Terminology: **Broker.**

Proxy Class ↔ between two systems → all good.

↳ but what about multiple systems?

Brokers are used then ← not too different from a proxy class.

Client → Proxy → Broker ⟷ Broker → Real Subject.

Only the Broker Class knows that an object on another system is being called

Secure Objects:

So if you remember the objects we were calling in our example.

Check the identity for who is calling the ~~Broker~~ Proxy Class, and then allow/reject a method call.

Date 20
M T W T F S S

In case of multiple proxy classes it will be more prominent.

Lazy Loading:

→ Something takes long.

→ lets say you have to load data from a data base. You load a portion of the data.

(If you remember when we were studying parse models) If file was too big we only loaded parts that were needed. [Partial loading of pages, etc].

On Database

[OPY-ON-WRITE

Dirty Red: lets say a transaction is writing something & another transaction reads it. The 2nd transaction wouldn't always have the correct value.

[How did we fix that in data bases?]

Clients reading → No issue.

Client writing → ??? → Real Subject Copy.

↓

Commit!

→ Reference Counter

who is calling the destructor..

Date ____ 20____
M T W T F S S

Final Content

→ Visual Studio Command Prompt.
(run as administrator).

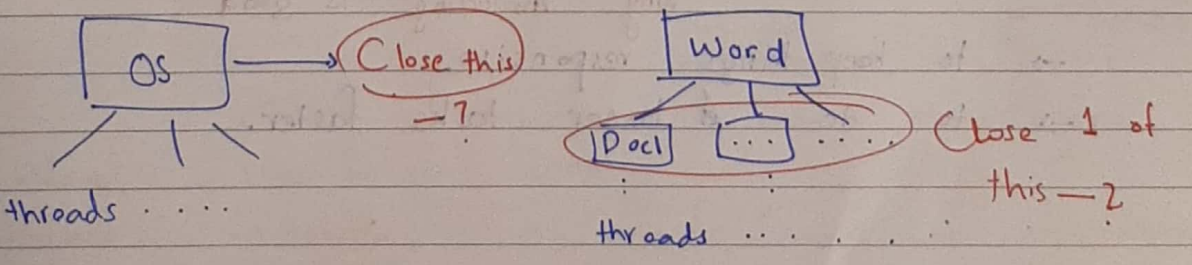
wsdl URL for webservice address /out : D:\name.cs
↓
is a proxy class

"THREADING IN C# & .NET"

- Every application has its processes.
- Different types of scheduling algorithms are used to manage processes [Recall]
  - Round Robin
  - Most frequently used, etc
- Processes tab on your task manager.

→ Why do we want to thread?
So we can run things in parallel.

→ A process is divided into threads.
→ A thread is a very small unit/part/chunk of a process.
→ Should not be highly dependant on the main thread

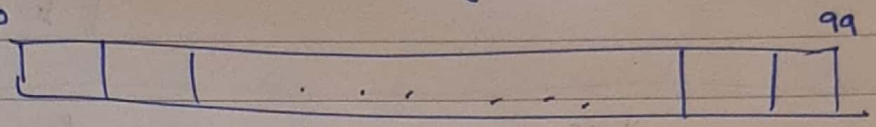


Normally when you are making an app, its single-threaded by default.

60 sec → divide into 6 thread-2

10 sec each.

Example of Threading:

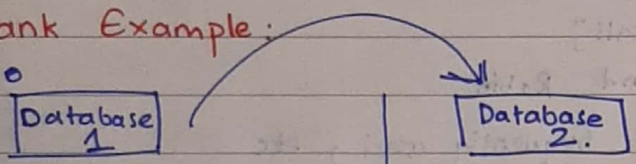


Make chunks of 10 elements of array.

≈ appox time ... $\frac{60}{10}$ (+ some more).

because some things are going on in the background. like context switching etc.

Bank Example:



5 millions of entries of data per day.

So 10 years worth of data

Solution? Make threads for some specific time periods.

facebook → Suggestion algorithms (recommendation) → Millions of people → thread to make faster.

Why Threading is Good

→ to keep things responsive.

→ to do longer tasks faster.

Date 20
M T W T F S S

Problems with Threading:

- Deadlock.
  - ↳ Based on read & write locks.
  - → That's why we need to lock threads.
- Race Condition.
  - → when two threads access a shared variable.
  - Race —? which thread writes first.

~~~ we can use threading for preprocessing arrays.

IO, Delegates, System.Threading
- { is automatically created
- } events (onClick() etc.
- ↳ If you want to make threads on your own.

Current Thread → What thread is currently Running?

Priority —? Higher Prio => more time

---

in main thread.

Thread t1 = new Thread (new ThreadStart (WorkThreadFunction));

(not main)

- ThreadStart → have to pass this as an object
- WorkThreadFunction → Delegate of your function

- Join ( ) → waits till worker threads are done.
- Sleep ( ) → Sleep for some time.
- Suspend ( ) → Resume ( ) → Old methods. Kind of like pause/resume
- Interrupt ( ) → immediately calls, as compared to wait for something.
~~~

Date 20
M T W T F S S

: Synchronization :

Locks:

lock keyword

- you can lock an object
- you use { } after lock (obj).
- { → starts lock
  } → releases lock.

Monitor: Similar to lock for compilers.

Monitor.Enter(obj) → starts the { (basically lock
Monitor.Exit(obj) → same as } (releases lock
You HAVE to use the try & finally blocks here.

Locks came first, then Monitor.

↳ PulseAll → notifies all threads before releasing.

Can't do this with Locks

Mutex Class

- Same things as lock & Monitor
- Doesn't see much use.
- Monitor has more functionality than Mutex

Interlocked Class:

[let's say you want to keep count of threads].

+ 1 when created.
− 1 when deleted

Interlocked Class: Is a helper class. lets us do this.

} you can also swap & compare.

But you can also do this with locks.

locking on a function level.

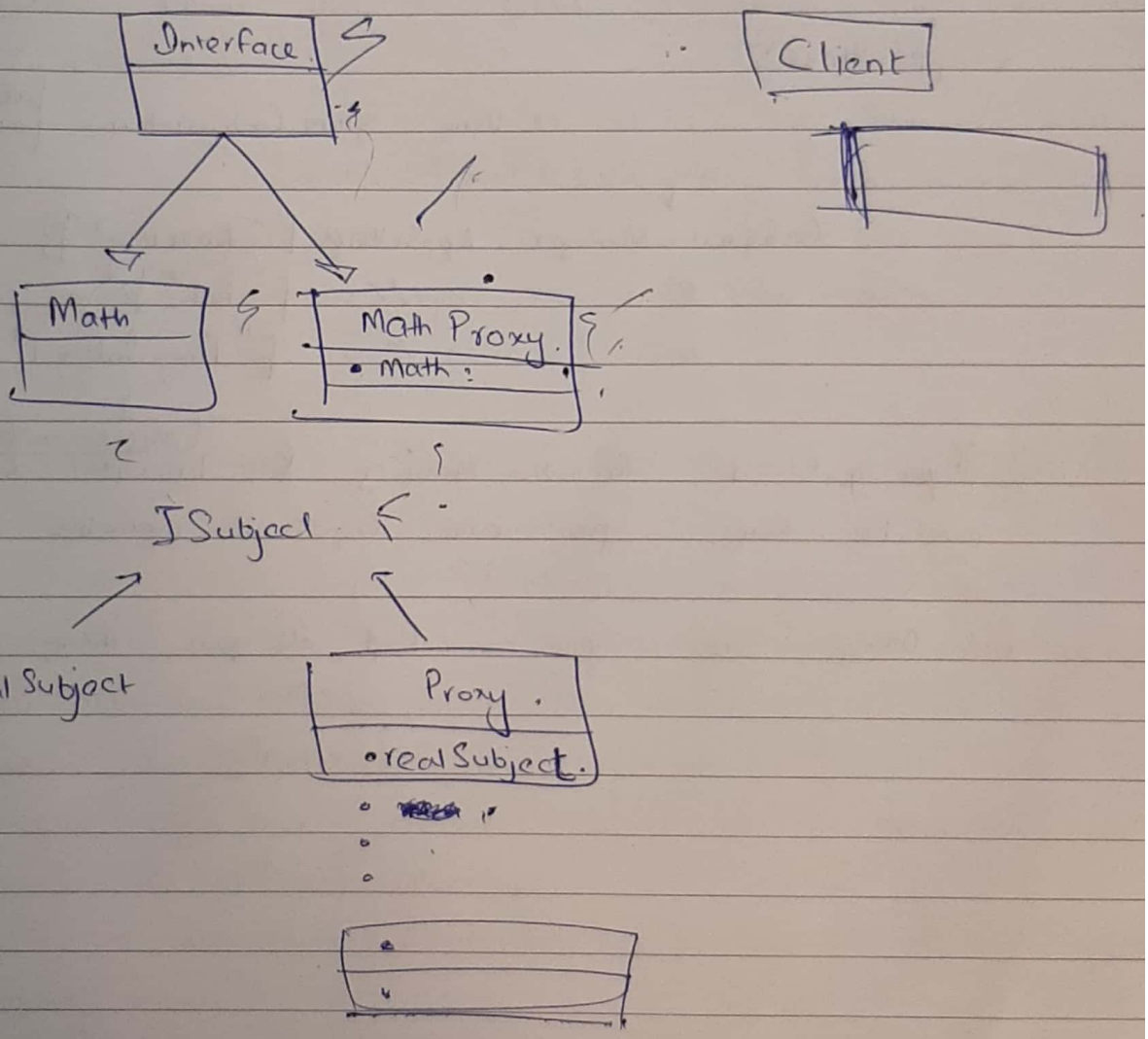
[MethodImpl] → Makes a method sequential.

↓ Only one thred can access at a time

[MethodImpl (MethodImplOptions.Synchronized)]

{

different things can go here.



WCF (continued)

Date 20
M T W T F S S

Config files :

When you create any application, VS makes an app.config file for you.

Adds some things by default.

You have to manually add the <appSettings> tag.

```
<appSettings>
    <add key = "Directory" value = "D:" />
    <add key = "Path" value = "D:\My folder" />
    <add key = "TimerInterval" value = "1000" />
</appSettings>
```

→ Using System.Configuration - [add your .dll by including a reference]

Static Class.

```
ConfigurationManager.AppSettings["Directory"];
                    //  ["Path"];
                    //  ["TimerInterval"];
```

{ might need to cast. }

If you just do Cofiguration Manager; & let it display the available things, you can see all available options.

in Debug folder → you can find all your things.

- .exe
- .config.

Date 20
M T W T F S S

Reflection in .NET

Not used all the time, but sometimes you end up needing it.

Is a more advanced C# topic.

Exists in C# & in other languages too. [Java]. e.g.

we have dlls in C# which was an assembly.

- Assembies had: (dll or exe).
  - ↳ ~~Meta~~ metadata
  - ↳ Intermediat Language.

Reflection allows you to gather information from objects/etc from runtime

- information → data from an object
  - ↳ let's say: items from a list.

What's different? You can also fetch method names

~~Ref~~ Reflection: Reads

at runtime → any object that is in the memory [data & functions].

You're reading the metadata from the assembly [dll/ex

- Object Information
  - ↳ about its structure/methods.
  - ↳ Now you can dynamically invoke functio

→ You can also create objects at runtime. (not by conditions & "new" statements).

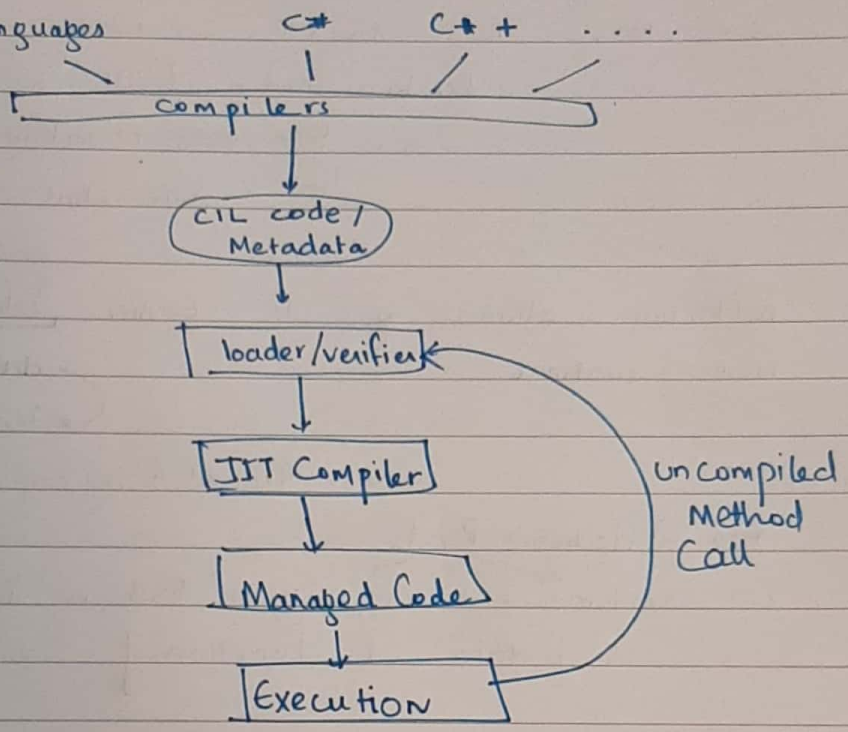
→ Maybe you have different copies of a program running for different clients. Now if a new client comes, you will have to build you program again.

→ You won't have to do this if you do it dynamically (and not via switch cases).

Date 20
M T W T F S S

Whenever you will create an object, you can call public methods. (Like you normally would).

.NET Executio Model.



METADATA

→ Every relevant information is stored in the metadata in a single location

→ Metadata for "types" → like Strings, int, or object of any class.

"Type of" method that we use to find types.

There exists a class named "Type"

Date 20
M T W T F S S

"Type"

In C++ we referred to the same things as attributes & properties.
But in C# we had discussed that attributes & properties are not the same.

Properties
- with get/set.

Fields
- all local variables.

```
public string Name{
    get { return name; }
    set { name = value; }
}
```

Normally you don't have to include System.Reflection

Member Info
- Is a super class that has multiple classes related to things we need.

Namespaces exist in an assembly.
- you can look at everything within an assembly.
- you cannot look at things not in the assembly.

Late Binding: dynamically loading objects at run-time & dynamically deciding which function to call.

The dynamic variables are useful for this.

→ You can create objects at runtime too.

Obfuscate : hiding some things, so that the code can't be read by Reflection.

[Skype Example | Windows Example].

Some alternates are available.
But most things are obfuscated in this case.

Ms cor lib.dll → Is installed with visual Studio
Is available in the Global Assembly Cache [most assemblies are here].
You load the assembly to the memory.

Assembly will have a namespace.

Date _____ 20___
M T W T F S S

REST v/s SOAP

↓

→ WCF Restful Services

↓

then was made into "Web API"

→ when you send a soap request, the message is sent in a soap envelope.

→ Body had the request details.

↳ asmx of WCF web service.

[↑ bt of information is being sent].

So instead sending that whole message, we just send that through URL

Advantage: You're passing less data.

More Parameters? You can extend the URL!

GET : You can pass information in the URL

Misconception: REST Service is the return message in JSON

What it is actually:

→ Every Service has its URL

Every module, every function

→ Use a protocl whose methods you will use.

Like HTTP GET & POST for HTTP Protocols.

- Do not make your own methods.

Example:

Uniquely Identifiable Resource.

/ user details /

↳ GET, DELETE, UPDATE

POST

Date 20
M T W T F S S

In WCF we had multiple protocols

HTTP TCP

But here, he says that your response should be based on what you are interacting with.

browser → HTML
Java Script → JSON
Java App → xml.
C# " → "

Client will inform what language they will communicate in.

→ Now different types of serilizations will be going on.

→ content-type → will be deciding to what language.

REST services should be stateless.

→ The URL type is going to decide whether you can access something.
→ authentication will only be done once. (while logging on).
→ You can pass authentication ID in the HTTP headers. Or through tokens.

Remember that if the server is keeping the information, it will become stateful

Date 20
M T W T F S S

RESTFUL ARCHITECTURAL STYLE

There is a concept of caching

→ When you're doing a HTTP request, you assume the file hasn't been changed if it has the same name:

"Some Script.js"

& the last modified date is the same. in the HTTP header.

If ~~the~~ the last modified is not specified, then you won't know if file has been updated ~~until~~ with just the file name, until the file name has been changed.

Load-balancing: when there's multiple servers, you try to equally divide the load.

Maintaining States... what if the server with the states fail-?

WADL → is an xml file format → a way to describe the behavior of HTTP resources.

(WSDL is a way to describe SOAP based web services.)

Web Application Description Language.

Is an xml description of HTTP based web services.

⇒ ADO.NET

↳ All services will be fetching data from the databases.
↓
ASMX
Web
Windows } services.

ADO used to be something that can fetch data from the databases. | Data Connectivity. || ADO.NET due to .NET
→ It isn't SQL specific.

- Provides a set of classes & methods that we can use for connectivity to any databases. And also how to perform transactions.

Different types of Queries.
→ Inline | → Stored Procedures, etc.

Different types of Queries with parameters.
→ Maybe hardcoded
→ Maybe user input.

" SQL Injection " → what is that?
→ Inserting malicious queries through user input.

How to avoid that? Encryption || Sanitization || SQL Params.

Data Reader → Reads line by line. (like in filing).
Data Adapter → Reads all data & then dumps it to some memory location. (uses data reader in the background).

You can describe the type of input that you want. Or the size.

Date 20
M T W T F S S

Data Table : Like how we have tables in SQL.
Is very similar to that.
(Is .NET specific).
You can used DataTable Instead of list here.

- Was originally made to use with normal database.
- But can be used with files too.

Data Set: Set of datatables
[Since a database can have many tables]

-Connected Environment-

- Connection remains open while you are reading the database.
- A database can have a set of connections open.
- You can't have more than "x" connections open.

-Data is always up-to-date.
- But !! Limited Connections.

-Disconnected Environment-

- Data adapted opens up a connection, then fills up a DataSet
DataSet → has DataTables → Data Rows
Data Columns.

- All insertion/deletion etc takes place in offline mode.
- Connection will then be established to quickly update & close the connection.

Date 20
M T W T F S S

Built-In Classes.

→ for data Providers.

- SQL | Oracle | oleDB | odbc
  - oleDB → file system
  - odbc → MySQL

+ Same interface is used to implement these.
+ So all will have similar method names.

What do you need for a Connection

Can also put this in app.config

- A connection String.
- A user ID. } → to connect to a database with a username / password.
- A password. }
- A Data Source. ⟵ Database [Server]
- An Initial Catalog ⟵ Database [Name]
- [Integrated Security] = true if you want only users of a domain to access the database.
  domain / user → is sent to DB.

Errors | Pooling
- errors while connecting to DB
- connection pools

Command Object

→ Connection → Your connection to the database.
→ Command Type → InlineQuery or Stored Procedure [Is an ENUM].
→ Command Text → Contains Query or SP name.
→ Parameters → for SQL Queries

Output Types: Data in the form of Table . or "count" or a number : 10 rows updated , etc.

app.config.

Date

Methods.

→ Execute Non Query:
Informs how many rows are effected.
↓↓
for update / delete

→ Execute Scalar.
One row x One Column (like count).
then you use this

→ Execute Reader
returns 1-n rows of data.

LINQ Extension Methods

Normally you can make methods in languages & to meet your requirements.
| Is leap year | has 30days | Is Numeric, etc.

→ Normally you have to make a Utility class.

So how can you make changes so that you can view the methods through intellisence.

↘ extend the class & update methods in there - But now you're using your own class, not the original one.

→ My String extends String { using MyString.

Extension Methods, Solve this! Extending without inheritance

(this string s)
this ↓ part makes it so it extends the String class.

Date 20
M T W T F S S

→ Extending without inheritance.
→ Sealed Classes? No problem, can still make extension methods for it.
→ Must be at the top (The extension methods).
[top-level static class]
↳ Must be parent.

LINQ
Language Integrated Query

One type of Query instead of

from
where
select

→ because we are directly working on the data.

→ Is it good performance-wise?

Sorting Example =>

from
order. By value (descending) keyword
Select value.

You don't have to write a whole method for it.

Date ____ 20____
M T W T F S S

LINQ to ADO.NET.

all classes/table exist as objects

LINQ to XML

load xml file here ← XElement obj

IENumerable<XElement> employees = obj.Elements();

→ Lambda Expressions

You can perform complicated tasks through a small set of expressions.

cust.Where(———).Select

Delayed execution.

All data → where Clause → Select Clause

This is done in one location, and sequentially & done instantly.

where & select are done side by side.
Rows & columns are reduced side by side.

Advantages

- You learn one syntax.
- Strongly typed → errors on compile time, not run time [by using var]
- Intellisense. (suggestions based on data types).

Date 20
M T W T F S S

LINQ to SQL.

↳ Only LINQ to SQL but we do not want to restrict ourselves

Limitations:

→ 1 : 1 table representations.

→ So joins used to be difficult.