SQL Databases

- Database is a collection of information organized so it can be easily accessed, managed and updated
  - ACID
    - Atomicity – requires that each transaction be "all or nothing" if one part fails then the entire transaction fails
    - Consistency – ensures that any transaction will bring the database from one valid state to another
    - Isolation – ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially (one after the other)
    - Durability – ensures that once transaction is committed. It will remain so even in the event of crash/power outage
  - Different types of databases:
    - Flat file (text file) – plain text file
    - Extensible Markup Language (XML) – human and machine readable
    - Excel – stores data in tabular form with relationships
    - Relational Database – structure that recognizes relations among stored items of information
- Why databases
  - Eliminate or reduce data redundancy
  - Eliminate or reduce data inconsistency
  - Data standardization
  - Secure all data
  - Maintain data integrity
- Logical relationship
  - Logical relational data model – All data is represented in terms of tuples, group into relations
  - Types:
    - One to one – both tables can have only one record on either side of the relationship. Each primary key relates to only one (or no) record in the related table
    - One to many – the primary key table contains only one record that relates to none, one or many records in related table
    - Many to many – each record in both tables can relate to any number of records (or none) in the other table
- Referential integrity – means that the foreign key in any referencing table must always refer to a valid row in the referenced table. Ensures that the relationship between two tables remains synchronized during Updates/Deletes
- Normalization – process of organizing the columns (attributes) and tables (relations) to minimize data redundancy
- Denormalization – process of optimizing read performance of database by adding redundant data back
- Transactional vs Analytical databases
  - Transactional DB – allows for CRUD actions (create/read/update/delete) to be taken in a transaction way to make atomic changes

- Analytical DB – db of data which is generally read-only and used to store current and historical data for the purpose of mining, generating statistics, projections and otherwise analyzing it for a certain pattern or criteria

❖ Transact (T-SQL) – structure query language. Microsoft's (and Sysbase) proprietary extension of SQL. Used for querying, altering, and defining DB

❖ SQL statement categories
- Data Definition Language (DDL) – Create, Alter, Drop, Truncate
- Data Manipulation Language (DML) – Select, Insert, Update, Delete
- Data Control Language (DCL) – Grant / Revoke permissions
- Transaction Control Language (TCL) – Begin, Commit, End, Rollback

❖ Truncate Table
- Removes all rows from a table without logging the individual row deletions
- Resets identity counter
- Cannot truncate if referenced by Foreign Key constraint
- Cannot active triggers because operation does not log individual row deletion
- For tables with one or more of these characteristics use DELETE instead

❖ Difference between Truncate and Delete statements

Truncate	Delete
DDL statement	DML statement
Removes all records by deallocating data pages	Removes row by row
Minimal logging (Fast)	Logs all deletions (slow)
Resets identity to its starting seed value	Retains identity seed
Takes less transaction space	Takes more transaction space
Cannot be used with indexed views	Can be used with indexed views
Cannot activate triggers	Activates triggers

❖ Join types
- Inner join – select rows where there are matches in both tables
- Left outer join – select all rows in left table and matches in right table
- Right outer join – select all rows in right table and matches in left table
- Full outer join – select all rows from left and right table and their matches
- Cross join – produces the Cartesian product of the tables (tbl1 rows * tbl2 rows)
- Self-join – select a matches on the same table

❖ Sub Queries – query within another query used to return data that will be used in the main query
- Can be used with DML statements
- Rules:
  - Must be inside parenthesis
  - Must only return one column
  - Cannot use order by (can use group by)
  - If it returns more than one row can be used with IN operator
  - Cannot reference any value that evaluates (BLOB, ARRAY, CLOB, NCOB)
  - Cannot be immediately enclosed in a set function
  - "Between" cannot be used with Sub Query (can use Between within Sub Query)

- ❖ UNION and UNION ALL – Used to combine the result set of two or more select statements
  - Union selects only distinct values by default, to allow duplicates use "ALL" (UNION ALL) keyword
  - Must have same number of columns
  - Must have similar data types
  - Must be in the same column order
- ❖ Views – Virtual table based on result set of a select statement. Can contain rows and columns from more than one table
  - Advantages
    - Hide complexity – if you have query that joins several table or does complex logic/calculations you can code all of into a view. Then select from view like a regular table
    - Security mechanism – can select specific columns of tables and give permission to access only those columns. Allows you to only give user the data he needs
    - Can simplify supporting legacy code – if refactoring a table would break a lot of code; we can replace table with view
  - Disadvantages
    - Can be slow. Even slower if views are created from other views
    - Cannot use DML operations
    - It is an object so it occupies space
- ❖ Aggregate functions – returns a single value calculated from values in a column
  - AVG, SUM, COUNT, FIRST, LAST, MAX, MIN
  - Group by – group set of rows into summary by values of one or more columns (aggregate function must have a group by)
  - Having – pass condition after aggregate
- ❖ Scalar functions – returns a single value, based on input value
  - UCASE, LCASE, LEN, ROUND, NOW, FORMAT, MID
- ❖ Conditional Statements IF-ELSE and CASE – imposes conditions on the execution of a T-SQL
  - CASE provides condition (WHEN, THEN, ELSE) to any ordinary SQL command such as select or update
- ❖ Stored procedure (SP) – set of logical group of SQL statements which are grouped to perform a specific task
  - Benefits
    - Increases performance of DB by reducing amount of info sent to DB
    - Compilation is only required once when SP is created
    - SP enhances security (no direct access to tables and views required)
    - Can be tested independently
  - Disadvantages
    - Limited code functionality (not as robust as application code)
    - No debuggers
    - Requires more specialized skills
    - SP language may differ from one DB system to another
    - Some SP may have business logic in DB which should not be there
- ❖ User Defined Functions (UDF) – routines that accept parameters, perform actions such as complex calculations and return the result of that action as value (Can return single scalar value or a result set)
  - Benefits

- Allow modular programming – reusable
- Faster execution – less compile time by caching the plans and reusing (no need to be reparsed and re-optimized with each use)
- Reduce network traffic by doing complex calculations on server and returning less number of rows (data)

- Types
  - Scalar functions – returns single data value
  - Table value functions – returns a table data type
  - System functions – provided by SQL. Cannot be modified

❖ Difference between UDF and SP

UDF	SP
Must return value	May or may not return value
Only SELECT statements	Can have SELECT and DML
Only input parameters	Can have input and output parameters
No TRY CATCH	Can have TRY CATCH
No transactions	Can use transactions
Only table variables	Can have both table variables and temporary tables
Cannot call SP	Can call UDF
Can call, from SELECT	Can only be executed (EXEC)
Can be used in JOIN	Cannot be used in JOIN

❖ Constraints – used to specify rules for data in table
- NOT NULL, UNIQUE KEY, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT
- If there is a violation between data and constraint, action is aborted
- Can be specified when table is created or after (CREATE and ALTER)

Can be defined two ways:

- Column Level – specified immediately after column definition
- Table Level – after all columns are defined

❖ Indexes – Special lookup tables that DB search engine can use to speed up data retrieval
- Index is a set of pages (nodes) organized in B-Tree structure
- Created on columns (no image, text, varchar(max) types)

Two types:

- Clustered index
  - Physically stores rows of data on leaf nodes
  - Can be stored ascending or descending order
  - Only one clustered index per table
  - Table with clustered index is “Clustered table”
  - Table with no clustered index is a “Heap”
- Non-Clustered index

- Leaf nodes contain values from indexed columns and pointer to data rows rather than rows themselves
- Takes an additional step in order to locate actual data
- Possible to have more than one per table

Disadvantages –indexes slow down inserts and updates (which can become a really serious issue with locking) and cost disk space

- Unique index – one which no two rows are permitted to have same index key (duplicate values must be removed before unique index can be created)
- When to avoid indexes
  - Don't use on small tables
  - Tables that have frequent, large batch update or insert
  - Don't use on columns with high number of NULL values
  - Columns that are frequently manipulated should not be indexed
- Transaction Management – group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all tasks in the group successfully complete (if any fails, transaction fails)
  - Begin, Commit, End, Rollback
- Cursors – object used to manipulate data in a set on a row by row basis
  - DECLARE, OPEN, FETCH, CLOSE, DEALLOCATE
  - Temporary work area created in system memory
  - Implicit cursors are automatically created whenever DML statement is executed. For Insert, cursor holds data to be inserted. For update and delete, cursor identifies rows that would be affected
  - Explicit cursors defined by programmer for gaining more control over context area

  Disadvantages

  - Uses more resources because each time you fetch a row from the cursor, it results in a network roundtrip
  - There are restrictions on the select statements that can be used
  - Performance and speed is slow
- How to avoid cursors
  - Use SQL while loop – we can insert result set into temp table
  - User defined functions – cursors are sometimes used to perform calculation on result row set. Can be achieved with UDF instead
- Triggers – automatically executed when an event occurs in DB by DML (AFTER, ISNTEAD OF)
- Triggers vs SP

Triggers	SP
Only executed through event	Can be executed whenever
Cannot be scheduled	Can be scheduled
Cannot take input	Can take input
Cannot return values	Can return values
Cannot use transaction	Can use transaction

Cannot call from front-end	Can call from front-end
Triggers are for auditing work	SP for performing tasks

- Dynamic SQL – build SQL statements dynamically at runtime (more flexible)
- Common Table Expression (CTE)– temporary result set that is defined within the execution scope of a single select, insert, delete, update or create view statement. similar to derived table in that it is not stored as an object and lasts only for duration of query. Can self-reference and be referenced multiple times
  - CTEs are more like temporary views than anything else. When you look at the execution plan, you'll see that they are inlined into the query, not materialized and stored. I find, with the exception of recursion, they're more to make queries simpler to write than faster to run
    - Used to store result of a complex sub query for further use.
    - Used to create a recursive query.
- Temp Variable Vs Temp Table

Temp variable	Temp table
Can be used in UDF	Cannot be used in UDF
Cannot add indexes explicitly	Can add indexes explicitly
Scope is batch	Scope of local # is session which it is created
Dropped automatically	Dropped automatically when session ends
Can explicitly drop when batch execution completes	Can explicitly drop any time

- IEnumerable – exposes an enumerator which supports simple iteration over a non-generic collection
  - executes queries immediately as it does not make use of SQL features
  - great for working with sequences that are iterated in-memory
  - suitable for LINQ to Object and LINQ to XML queries
- IQueryable – provides functionality to evaluate queries against a specific data source wherein type of data is not specified
  - extension of IEnumerable built to specifically deal with SQL queries
  - allows for out of memory things like a remote data source, such as a database or web service
  - suitable for LINQ to SQL queries
- Rank() function – return the ranking set of values within a given partition
- DenseRank() – return the rank of rows within partition of a result set, without any gaps in the ranking
- Row_Number() – assigns unique number to each row within PARTITION given the ORDER BY clause
- @@ROWCOUNT – returns the number of rows affected by the last statement. If its more than 2 billion use ROWCOUNT_BIG
- SQL Profiler – is a graphical tool that allows system administrators to monitor events in an instance of Microsoft SQL Server. You can capture and save data about each event to a file or SQL Server table to analyze later
- How to improve query performance?
  - Parameterized Queries – If your application runs a series of queries that are only different in some constants, you can improve performance by using a parameterized query. (better performance by compiling the query only once and executed plan multiple times)
  - Eliminate cursors – remove and use set-based queries (more efficient). Cursors are slow

- Avoid the use of non-correlated subqueries – have query outside of main query and store result in a variable which can be used later anywhere
- Rewrite subqueries as joins - Sometimes you can rewrite a subquery to use JOIN and achieve better performance. The advantage of creating a JOIN is that you can evaluate tables in a different order from that defined by the query. The advantage of using a subquery is that it is frequently not necessary to scan all rows from the subquery to evaluate the subquery expression
- Create and use indexes – could improve performance when choosing proper columns to index (also the correct index type to use clustered or non-clustered)
- Avoid indexing small tables – more efficient to do table scan
- Drop unused indexes – if there's no retrieval being done it will improve data modification
- Create Highly selective index – Indexing on columns used in the WHERE clause of your critical queries frequently improves performance. However, this depends on how selective the index is. Selectivity is the ratio of qualifying rows to total rows. If the ratio is low, the index is highly selective. It can get rid of most of the rows and greatly reduce the size of the result set. It is therefore a useful index to create. By contrast, an index that is not selective is not as useful

ASP.NET

❖ ASP.NET is an open-source server-side web application framework designed for web development to produce dynamic web pages. Developed by Microsoft to allow programmers to build dynamic web sites, web applications and web services

❖ Characteristics of ASP.NET

- Web Forms – these pages can be written using a combination of HTML, client-script, server controls, and server code. When users request a page, it is compiled and executed on the server by the framework, and then the framework generates the HTML markup that the browser can render.
- Code-behind model – refers to code for your ASP.NET page that is contained within a separate class file. This allows a clean separation of your HTML from your presentation logic
- Directives – Directives specify settings that are used by the page and user-control compilers when the compilers process ASP.NET Web Forms pages (.aspx files) and user control (.ascx) files. User controls – encapsulations of sections of pages which are registered and used as controls
- Custom controls – build custom controls. Unlike user controls, these controls do not have an ASCX markup file, having all their code compiled into a dynamic link library (DLL) can be used across multiple web applications
- Rendering technique – uses a "visited composites" rendering technique
- State management – ASP.NET application is hosted by a web server and are accessed using stateless HTTP protocol. As such if an application uses stateful interaction it has to implement state management on its own
    - Application state – held by a collection of shared user defined variables
    - Session state – held by a collection of user defined session variables that are persistent during a user session
    - View State – refers to the page-level state management mechanism, utilized by the html pages emitted by ASP.NET application to maintain the state of the Web Form controls and widgets

- Server-side caching – ASP.Net offers a “Cache” object that is shared across the application and can also be used to store various objects. “Cache” object holds the data only for a specified amount of time and is automatically cleaned after the session time-limit elapses

❖ ASP.NET Extension

- AJAX – an extension with both client-side as well as server-side components for writing pages that incorporate AJAX functionality
- MVC – framework extension to author ASP.NET pages using model view controller architecture
- Razor – web pages’ view alternative to Web Forms designed for use with MVC
- Dynamic data – scaffolding extension to build data driven web applications
- Web API – HTTP API framework for exposing web services
- SignalR – real time communication framework for bi-directional communication between client and server

❖ ASP.NET Versions

- 3.0:
  - Windows Presentation Foundation (WPF)
  - Windows Workflow Foundation (WF)
- 3.5:
  - Integrated ASP.NET AJAX
  - Support LINQ
  - New Data controls LINQ DataSource, ListView and DataPager
  - Dynamic Data
  - Multi-targeting framework support
- 4.0:
  - Introduced ClientIdMode property for Server Control
  - Routing
  - Introduced Meta tags, MetaKeyword and Meta Description
  - Chart Control
- 4.5:
  - Strongly typed data controls
  - Model Binding
  - Unobtrusive validation
  - Bundling and Minification
  - Async support
  - Support for asynchronous modules and handlers
  - Friendly URL
  - HTML5 features and enhancements
  - Support for WebSocket protocol
  - oAuth support
- 4.5.1:
  - Released with VS 2012 for Server 2012, windows 8
  - Bootstrap 3.0
  - Web API 2: OAuth 2.0, OData improvements, CORS
  - MVC5 attribute routing, authentication filters and filter overrides
  - Entity Framework 6

    - OWIN
    - SignalR
- 4.5.2:
    - Released with VS 2013 for Server 2012 R2 and windows 8.1
    - Higher reliability HTTP header inspection and modification methods
    - New way to schedule background asynchronous worker tasks
- 4.6:
    - Released with VS 2015 for Server 2016 and windows 10
    - HTTP/2 support when running on windows 10
    - More async task-returning APIs

❖ Client server application - computer network diagram of clients communicating with a server via the Internet. The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients
❖ Client and server communication – exchange messages in request-response messaging pattern
❖ Statelessness
- Stateless means that a protocol (agreed upon set of rules) or application program keeps no information on sequence of interactions with user, computer, program, etc.
- Statelessness is the opposite of maintaining state. It offers the advantages that it can simplify programming and that it can reduce network traffic
- State is maintained by most modern application programs in order to ensure data consistency and facilitate ease of use. Major benefit is that it allows programs to remember what users were doing earlier in the session
- User Datagram Protocol one of the core protocols of internet is stateless

❖ Event handling – subroutine describing what to do when an event is raised
- Event is an action or occurrence such as a mouse click, key press, mouse movement or any system generated notification (a process communicates through events)
- Raised at the client machine and handled at the server machine
- When event message is transmitted to server, it checks whether the event has an associated handler and executes it
- Generally, takes two parameters and returns void (first parameter represents object raising event, second parameter is the event argument)

❖ Application and Session Events
- Most important Application events are:
    - Application_Start – raised when the application is started
    - Aplication_End – raised when the application is stopped
- Most used Session events are:
    - Session_Start – raised when user first requests a page from application
    - Session_End – raised when the session ends

❖ Page and Control Events
- DataBinding – raised when a control binds to data source
- Disposed – raised when page or control is released
- Error – raised when an unhandled exception is thrown
- Init – raised when the page or control is initialized
- Load – raised when the page or control is loaded
- PreRender – raised when the page or control is to be rendered

- Unload – raised when page or control is unloaded from memory

❖ State Management – HTTP is a stateless protocol. Once the server serves any request from the user, it cleans up all the resources used to serve that request. These resources include objects created, memory allocated, etc.
- We need state management to track user information between page visits
- ASP.NET state management is the process which let developers maintain state and page information over multiple requests
- Mainly two types of State Management: Client-side and Server-side

❖ Client side state management
- View state – collection of key-value pairs, where key is string and value is object. Only those objects which are marked as Serializable can be stored in ViewState
  - Hidden element in form tag rendered
  - It is different for different clients and is submitted to server only if form is submitted or posted back
  - Cannot be programmed on client in JavaScript because the value of it is encrypted before it is rendered to browser
  - If ViewState is disabled, anything added to it is not retained
  - ViewStateMode can take three values – enabled, disabled, inherit
- Hidden fields – used for managing state of client in round trips to the same webform
  - Can store string data and cannot store serializable custom objects unlike ViewState
  - Value can be programmatically set on client using JavaScript, the same can be retrieved on server from HiddenField object
- Cookies – name value pair which on behalf of server is saved on client machine (memory or file)
  - Web browsers automatically include cookies along with every request it submits to server
  - Used for sharing data across webforms request by same client

  Types of cookies:
    - Persistent Cookie – are permanently stored on client machine in a file. Shared by all browser instances of the same type (Firefox, IE, Chrome, etc) running on machine
    - Non-Persistent Cookie – also called Session Cookie; temporarily stored in browser's memory. Not shared across browser instances. Every instance has its own set of non-persistent cookies
- Query strings – is the string in URL after "?".
  - Collection of key-value pairs (both are string type)
  - Ideal for transferring data from one page to other when using HyperLink or Response.Redirect
  - Should not have special characters like "&, +, #" etc. used in parameter or value

❖ Server side state management
- Application state –global storage mechanism that is accessible from all pages in the web application.
  - Stored in the application key/value dictionary
  - This information is available to all users of the website (if we need user specific information, it is better to use session state)

  Three application events:
  - Application_Start (use to initialize application variables)
  - Application_End (use to perform logging)
  - Application_Error (use to perform error logging)
- Session state – an object (HttpSessionState) created on server on behalf of given client

    - Object is created upon server receiving the first request from client and same object is reused for all subsequent requests by client
    - Has key value pairs representing the state of the client on behalf of whom it is created
    - Key value pair is of type string and object. We can store any type of data unlike Cookies where we can only store strings
    - Can store any amount of data and also we can have unlimited number of key value pairs
    - Time out period is defined for every session (default 20 minutes)
    - Can be destroyed programmatically using Session.Abandon() (destroyed after response of current page is rendered. Not immediately when line of code is executed)
    - Session.Remove() removes all keys in the session but session object is not destroyed
- ❖ Master Page (.master) – allows you to create a consistent look and behavior for all pages (or group of pages) in application
    - Cannot be directly accessed from the client because it just acts as a template for content pages
    - We can have content either inside ContentPlaceHolder or outside it (only content inside can be customized in content page)
    - Can have multiple master pages in one application
    - Can have a Master page be master of another Master Page
    - Content pages can only be placed inside content tag
    - Controls of Master Page can be programmed in Master Page itself (control in content page will never be programmed in master page)
    - Master Page of one application cannot be used in another application
    - Order of events raised – Load (Page) > Load (Master) > LoadComplete (Page). If we want to overwrite something done in Load, we have to code it in the LoadComplete event
- ❖ User Control and Custom Control – create your own controls
    - User controls (.ascx) – are containers into which you can put markup and web server controls. You can then treat the user control as a unit and define properties and methods to it
        - Easier to create than custom controls, because you can reuse existing controls
        - Instead of @Page directive it has @Control directive that defines configuration and other properties
        - Cannot run as stand-alone files (must add them to ASP.NET pages)
        - Does not have html, body or form elements in it (must be in host page. When registering it. .ascx file, tag prefix, tag name)
    - Custom controls – class that you write that derives from Control or WebControl
- ❖ Validation Controls
    - RequiredFieldValidator – ensures that the required field is not empty
    - RangeValidator – ensures input value falls within predetermined range
    - CompareValidator – compares a value in one control with a fixed value or value in another control
    - RegularExpressionValidator – allows validating the input text by matching against a pattern
    - CustomValidator – allows writing application specific custom validation routines for both client and server side validation
    - ValidationSummary – shows a summary of all errors in the page. Displays the values of ErrorMessage property of all validation controls that failed
- ❖ Validation Groups – group validation for information provided in different panels of complex pages
- ❖ AJAX control (Asynchronous JavaScript and XML) – cross platform technology which speeds up response time. AJAX server controls add script to the page which is executed and processed by browser

- Ajax.NET is AJAX framework integrated with ASP.NET framework
- AJAX returns a Promise (.done(), .fail(), .always(), .then())
  Controls:
  - ScriptManager – must be preset on every page which is using other AJAX controls
  - UpdatePanel – Container for fragment of the page which should be asynchronously posted when required
  - UpdateProgress Container for showing something when asynchronous postback is in progress
  - Timer – refreshing a fragment of the page at a regular interval of time

❖ Basic Controls
- Button – displays text within a rectangular area
- Link Button – displays text that looks like a hyperlink
- Image Button – displays an image
- Text Box – used to accept input from user. Can accept one or more lines of text depending on attributes
- Label – provide an easy way to display text which can be changed from one execution of page to next (use regular text if you want to display text that does not change)
- Check Box – displays single option that user can either check or uncheck
- Radio Button – only one option can be selected
- List Controls (drop-down list, List box, Radio button list, Check box list, Bulleted list). These controls let user choose from one or more items from a list
- HyperLink – like HTML <a> element
- Image - used for displaying images on the page

❖ Data Bound Controls
- GridView – allows for tabular data display and editing using a declarative approach (successor to DataGrid)
  - Binding to data source (SqlDataSource)
  - Built in sorting capabilities
  - Built in updating and deleting capabilities
  - Build in paging capabilities
  - Built in row selection capabilities
  - Programmatic access to object model to dynamically set properties, handle events, etc.
  - Multiple key fields
  - Multiple data fields for hyperlink columns
  - Customizable appearance through themes and styles
- DetailsView – used to display a single record from a data source in a table, where each field of record is display in a Row of table (can be used in combination with GridView)
  - Binding to data source controls (SqlDataSource)
  - Built in inserting capabilities
  - Built in update and deleting capabilities
  - Build it paging capabilities
  - Programmatic access to object model to dynamically set properties, handle events, etc.
  - Customizable appearance through themes and styles
- SiteMapPath – provides a bread crumb navigation control. Can be easily data bound to hierarchical data sources such as SiteMapDataSource or XmlDataSource

❖ Stream
- FileStream – used to read from, write to and manipulate file-related operations. Supports random access to files using seek method

- MemoryStream – reads or writes to an array of bytes in memory. If you create a MemoryStream with no constructor arguments, you get a byte array that automatically expands when needed
- StreamReader – reads character from streams, using encoding to convert characters to and from bytes
- StreamWriter – write characters to stream, using encoding to convert characters to bytes
- BinaryReader/BinaryWriter – read and write encoded strings and primitive data types from and to basic stream

❖ Globalization – process of designing and developing applications that function for multiple cultures
❖ Localization – process of customizing your application for a given culture and locale
❖ NuGet – package manager for Microsoft development platform. Provides ability to produce and consume packages
❖ Garbage collection – CLR has garbage collector that executes as a part of our program and responsible for reclaiming memory of no longer used objects
- If heap does not have sufficient memory to allocate a requested object, garbage collection will occur
- Garbage collector is given lowest priority. But when there is no space in heap, it is given real-time (highest) priority

❖ Page Life Cycle – SILVER
- Start – where the page properties such as Request, Response, IsPostBack and UICulture are set
- Initialize – themes are applied and unique ids are generated and set for controls (Init, InitComplete, PreLoad)
- Load – controls are loaded with information retrieved from view and control states. (OnLoad) set properties for all of the server controls of page, request query strings, and establish DB connections
- Validation – controls that require validation are validated here
- Event Handling – even handling for server controls (Click, SelectedIndexChanged, etc) are applied to server controls and fired by the control
- Render – page object calls this method on each control to write out HTML markup for the control to the browser

❖ Different types of caching
- Output caching – ability of the web server to cache a certain webpage after user request in its memory so that further requests for the same page will check for the cached page's validity and will not result in resource usage (DB access or file access) and the page will be returned to user from cache
- Application caching – mechanism for storing the Data objects on cache. ASP.NET allows us to store the object in a Key-Value based cache. We can use this to store the data that need to cached
- Partial page caching – we need to create custom user controls in order to achieve partial page caching

ADO.NET

❖ ADO.NET – provides a bridge between front end controls and back end database. ADO.NET objects encapsulate all data access operations and the controls interact with these objects to display data, thus hiding details of data movement
❖ DataSet – represents a subset of the database (contains DataTable and DataRelation objects)
- Does not have a continuous connection to the database
- To update the database a reconnection is required

- In memory representation of DB
  - DataTable – represents the tables in the DB
  - DataRow – represents a row in a table
  - DataAdapter – acts as a mediator between the DataSet object and DB. Helps the DataSet to contain data from multiple DB and other sources (used to fill a DataSet/DataTable with query results)

Advantages
  - Dataset can contain more than one table and we can create relationships among those tables
  - We can read the data by for each loop
  - Dataset can contain data from different data sources like oracle, sqlserver, access, etc

Disadvantages
  - Data is loaded into DataSet if other Client updates data same time in DB server the updated data will notreflect automatically in DataSet.
  - Slower in comparison to DataReader

❖ DataReader – provides a connection oriented access to the data records in the database. Suitable for read-only access, such as populating a list and then breaking the connection
- Access the results sequentially from DB
- Used to get forward only sequential results as query executes
  - DbConnection – represents a connection to the data source
  - DbCommand – represents the command or stored procedure sent to DB from retrieving or manipulating data

Advatanges
  - Faster, less memory – in control of how data is loaded, can get specific data and populate your own collection (no need for data table)

Disadvatanges
  - Must keep connection open and close connection after use

❖ SqlCommand – tell database about the operation we need to perform
- Select command – return a set of rows
- Insert command – return the number of rows inserted
- Delete command – return number of rows deleted
- Update command – return number of rows updated

❖ ExecuteReader – sends command text to the connection and builds SqlDataReader
❖ ExecuteScalar – returns first column in the result set returned by query. Additional columns are ignored
❖ ExecuteNonQuery – return value is the number of rows affected by the command for (insert, update, delete)
❖ SqlParameters – allows us to pass values into queries, sp, etc as parameters

Entity Framework (EF)

❖ Entity Framework (EF) is an object-relational mapper (ORM) that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that need to be written
- Retrieve and manipulate data as strongly typed objects

Advantages of ORM
  - Concurrency, cache and transaction management
  - Strong type safety
  - Focus on coding

- Better business modeling

Disadvantages of ORM

- Another abstraction layer
- More learning
- Can be slow

❖ Advantage of EF

- Provides auto generated code
- Reduce development time
- Reduce development cost
- Enables developers to visually design models and mapping of database
- Provides capability of programming a conceptual model
- Provides unique syntax for all object queries whether it is database or not
- Allows for multiple conceptual models to be mapped to a single storage schema
- Easy to map business objects (drag and drop tables)

❖ Disadvantage of EF

- Lazy loading is the main drawback of EF
- Syntax is complicated
- Logical schema is not able to understand business entities and relation among each other
- Logical schema of database is not capable of using certain parts of application
- Not available for every RDMS
- Need to handle data in a non-traditional way
- Does not work if we change any schema of the database. Must update the schema on the solution
- Not good for huge domain model

❖ Usage of EF

- Database First Approach
  - Load anything you need into the designed (tables, functions, SPs, etc)
  - System generates classes mapped to tables
  - Can interact with data using context object
- Code First Approach
  - Design your objects
  - Create a migration plan
  - Create the database structure, constrain and data
  - Can interact with data using context object

❖ Lazy loading –is a design pattern to defer initialization of an object until the point at which it is needed. It can contribute to efficiency in the program's operation if properly and appropriately used.

- The opposite of lazy loading is eager loading.

Advantages

- Minimizes start up time of the application.
- Application consumes less memory because of on-demand loading
- Unnecessary database SQL execution is avoided

Disadvantages

- need to do checks if the loading is needed or not, there is a slight decrease in performance

Language-Integrated Query (LINQ)

❖ Language-Integrated Query (LINQ) is a subset of instructions very similar to T-SQL that allow programmer to query any IQueriable object

- LINQ expressions are not executed immediately
- Method based syntax is totally equivalent to expression based syntax
- Types of LINQ – LINQ to Objects, XML, DataSet, SQL, Entities
- Can use lambda expression

Advantages
  - Quick turnaround for development
  - Queries can be dynamic
  - Tables are automatically created into class
  - Columns are automatically created into properties
  - Relationship are automatically appeaded to classes
  - Lambda expressions are awesome
  - Data is easy to setup and use

Disadvantages
  - No clear outline for Tiers
  - No good way of view permissions
  - Small data sets will take longer to build the query than execute
  - There is an overhead for creating queries
  - When queries are moved from sql to application side, joins are very slow
  - DBML concurrency issues
  - Hard to understand advance queries using Expressions

❖ Architecture – has 3 layered architectures in which uppermost layer consists of language extensions and bottom layer consists of data sources that are typically objects implementing (IEnumerable<T> or IQueryable<T> interfaces)

❖ Lambda expression (=>) – are basically anonymous delegates constructed by the Expression class to provide simple function usage without detailed implementation

❖ LINQ with Lambda
- lambda in LINQ to produce more readable and elegant code
- Provides a standard way to query results with a very low learning curve
- Reduces cruft (no explicit delegate declaration and function creation for simple one time use cases, and no dealing with Expression class directly)

❖ LINQ to SQL – offers an infrastructure (run-time) for management of relational data as objects. Translates language integrated queries to SQL and translates again after obtaining result to objects

❖ LINQ to Entities – does not have the limitations of LINQ to SQL that allows data query only in SQL server DB. Facilitates data query in a large number of data provides like Oracle, MySQL, etc.

Software Development Life Cycle (SDLC)

❖ Software Development Life Cycle (SDLC) – process used to design, develop and test high quality software. Aims to produce high quality software that reaches completion within times and cost estimates

❖ Waterfall – sequential design process (conception, initiation, analysis, design, construction, testing, implementation, and maintenance)
- Advantages
  - Stresses record keeping (documentation)
  - Client knows what to expect (size, cost, timeline, end result)
  - In case of employee turnover, strong documentation allows for minimal project impact
- Disadvantages
  - Once stage has been completed, cannot go back and make changes

    - If error is found in requirement or needs change. Project has to start from beginning with all new code
    - Only tested at the end. if bugs are found late, their existence may have affected the rest of code
    - Doesn't take into account client's evolving needs. If client realizes that they need more than they initially thought, and demand change. Project will come in late and impact budget
- When to use
    - There is a clear picture of what final product should be
    - Clients won't have the ability to change the scope of project once it has begun
    - Definition is success, not speed

❖ Agile – conceptual framework that promotes development iterations throughout the life-cycle of project

- Software developed during one unit of time is referred to as an iteration (may last from one to four weeks)
- Emphasize working software as the primary measure of progress

Characteristics:

- Light weight methodology
- Small to medium sized teams
- Vague / changing requirements
- Vague / changing techniques
- Simple design
- Minimal system into production

Advantages

- Allows for changes to be made after initial planning
- Easier to add features that will keep the project up to date with latest technologies
- At the end of each sprint, project priorities are evaluated. Allows clients to add their feedback so the ultimately get product they desire
- Testing done at the end of each sprint ensures that bugs are caught and take care of in the development life cycle
- Product can be launched at the end of any cycle (more likely to reach its launch date)

Disadvantages

- Depends on project manager being good (otherwise project may come in late and over budget)
- Initial project doesn't have definitive plan, so the final product can be very different than what was initially intended

When to use

- Rapid production is more important than quality of product
- Clients can change the scope of project
- There isn't clear picture of what final product should look like
- You have skilled developers who are adaptable and able to think independently

- Product is intended for an industry with rapidly changing standards

❖ Different Agile Methodologies

- Scrum – focuses on maximizing team's ability to deliver quickly, to respond to emerging requirements and to adapt to evolving technologies and changes in market condition
- Extreme Programming (XP) – intended to improve software quality and responsiveness to changing customer requirements. Frequent releases in short cycles to improve productivity and introduce checkpoints at which new customer requirements can be adopted
- Lean – 7 principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, see the whole
- TDD – test first development. Write tests before you write production code to fulfill test and refactoring

❖ Scrum

- Planning – beginning of a sprint, team holds a planning
  - Communicate scope of work during the sprint
  - Select product backlog items that can be done by voting
- Stand-up – 15 minutes every day at same time during sprint, team holds daily stand-up meeting
  - Anyone can contribute (not only scrum team)
    Each team member must answer three questions
    - what did I do yesterday that helped meet sprint goal?
    - What will I do today to help meet sprint goal?
    - Do I see any impediment that prevents me or team from meeting sprint goal?
- Review and retrospective – at end of sprint team holds review and retrospective

  Reviews – 2 hours is recommended duration for two-week sprint

  - Work that was completed and the planned work that was not completed
  - Presents the completed work to stakeholders (demo). Incomplete work cannot be demonstrated

  Retrospective – 1:30 hours is recommended for two-week sprint

  - Reflect on past sprint
  - Identify and agree on improvement actions
  - Facilitated by scrum master

  Two questions

  - What went well during the sprint?
  - What could be improved in the next sprint?

❖ Difference between Scrum and XP

SCRUM	XP
Typically sprints from two weeks to one month long	Typically iteration from one or two weeks long
Does not allow changes into their sprints. Once sprint planning meeting is completed and	As long as team hasn't started working on a particular feature, a new feature of equivalent size can be swapped into the iteration

committed to develop a set of product backlog items	
Scrum owner prioritizes product backlog. But team decides what to develop next	Work in strict priority order. Features are prioritized by customer
Doesn't use any engineering practices	XP uses engineering practices such as TDD, Pair programming, simple design, etc

- ❖ Team Foundation Server (TFS) – source code management, reporting, requirement management, project management, automated builds, lab management, testing and release management capabilities
  - Version control – store and collaborate on code with unlimited private repositories (uses Git)
- ❖ Software Architecture – process of defining a structured solution that meets all of the technical and operational requirements, while optimizing quality attributes such as performance, security, and manageability. Involves a series of decisions based on a wide range of factors that have considerable impact on quality, performance, maintainability and success of application
  - Goals of architecture:
    - Expose the structure of the system but hide the implementation details
    - Realize all of the use cases and scenarios
    - Try to address the requirements of various stakeholders
    - Handle both functional and quality requirements
- ❖ Model-View-Controller (MVC) – software architectural pattern mostly for implementing user interfaces. Divides a software into three interconnected parts to separate internal representation of information from the way information is presented to user
- ❖ N-Tier – client-server architecture in which presentation, application processing, and data management are physically separated. Most used multitier architecture is three-tier architecture
  - Presentation tier – top most level, user interface. Translates tasks and results to something users understand
  - Logic tier – coordinates the application. Processes commands, makes logical decisions/evaluations and performs calculations. Also moves and processes data between the two surrounding layers
  - Data tier – information is stored and retrieved from database or file system. Information is then passed to logic tier for processing and eventually to user
- ❖ Service Oriented Architecture (SOA) – architectural pattern in which application components provide services to other components via communication protocol, typically over a network
- ❖ Design patterns
  - Factory pattern
  - Abstract factory pattern
  - Behavioral pattern
  - Singleton pattern

C# and Object Oriented Programming (OOP)

- ❖ C# is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object oriented, and component-oriented programming disciplines

  Features

- Unified object system – everything is an object including primitives
- Single inheritance – classes have only one base class
- Interfaces – specify methods and interfaces, but not implementation. Contains only signatures of methods, properties, events, indexers. Class or Struct can implement interface and must include the members in the interface
- Abstract class –cannot be instantiated. can contain either abstract methods or non-abstract methods. Abstract members do not have any implementation in the abstract class
- Structs – a restricted, lightweight (efficient) value type that is typically used to encapsulate small groups of related variables
- Delegates – expressive typesafe function pointer. Useful for strategy and observer design patterns. Allows programmers to encapsulate a reference to a method inside a delegate object. Delegate object can then be passed to code which can call the referenced method without having to know at compile time which method will be invoked
- Preprocessor directives – gives instruction to the compiler to preprocess the information before actual compilation starts (all preprocessor directives begin with #)

❖ Interface vs Abstract class

Interface	Abstract class
A class may inherit several interfaces	A class may inherit only one abstract class
Cannot provide any code, just signature (no implementation)	Can provide code (implementation)
Can only be public	Can have different access modifiers
Interfaces are used to define the peripheral abilities of a class. In other words both Human and Vehicle can inherit from a IMovable interface	An abstract class defines the core identity of a class and there it is used for objects of the same type
If various implementations only share method signatures then it is better to use Interfaces.	If various implementations are of the same kind and use common behavior or status then abstract class is better to use.
Requires more time to find the actual method in the corresponding classes.	Fast
If we add a new method to an Interface then we have to track down all the implementations of the interface and define implementation for the new method.	If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly.
No fields can be defined in interfaces	An abstract class can have fields and constants defined

❖ When to use abstract class and interface

- If you have a lot of methods and want default implementation for some of them, then go with abstract class
- If you want to implement multiple inheritance, then you have to use interface. subclass cannot extend more than one class but you can implement multiple interface

- If your base contract keeps on changing, then you should use abstract class, as if you keep changing your base contract and use interface, then you have to change all the classes which implements that interface.

❖ Types of comments
- /* lines of comments */
- // single line comment
- /// <comment_in_xml> automatic XML commenting facility

❖ Enumerations (enum) – distinct type that consists of a set of named constants
- Base type can be any integral type except for char
- Defaults to int
- Must cast to int to display in WriteIN

❖ Features of Struct
- Can have methods, fields, indexers, properties, operator methods and events
- Can have defined constructors, but no destructors
- Default constructor is automatically defined and cannot be changed
- Cannot inherit other Structs or classes
- Can implement one or more interfaces
- Cannot be specified as abstract, virtual or protected
- Can be instantiated without using the "new" operator
- If "new" is not used, fields remain unassigned and object cannot be used until all fields are initialized

❖ Exception handling – exception is a problem that arises during execution of a program
- Try – identifies a block of code for which particular exceptions is activated
- Catch – catches the exception and handles it
- Finally – always executes even if no exceptions are caught
- Throw – throws an exception when a problem shows up
- Throw ex – resets the stack trace so errors you would see only originate from HandleException

❖ Object oriented application – it's a collection of related objects communicating with each other in a controlled environment as per rules of the business

❖ Classes and Objects
- Class combines together class variable (data) and methods (behavior)
- Instantiation of class (creating objects) is required to use variables and methods

❖ Namespaces – designed for providing a way to keep one set of names separate from another. The class names in one namespace do no conflict with same class names in another

❖ Metadata – information that is stored about your program

❖ Reflection – are objects used for obtaining or dynamically adding type, values, and object information at runtime. Classes that give access to the metadata of a running program

❖ Encapsulation – process of enclosing one or more items within a physical or logical package
- Prevents access to implementation details
- Implemented by using access specifiers
- Types of access specifiers
  - Public – can be accessed by everyone outside class
  - Private – can only be accessed within current class

    - Protected – can be accessed within current class and child classes
    - Internal – can be accessed in the current assembly
    - Protected internal – can be accessed by child class within the same application
- ❖ Inheritance – process of acquiring the existing functionality of the parent class
    - Advantages are generalization, extensibility, reusability
- ❖ Polymorphism – object may take on different forms and in each form it exhibits the same functionality but implemented in different ways
    - Static – response to a function is determined at the compile time (method overloading)
    - Dynamic – decided at run-time (method overriding)
- ❖ Operator overloading – permits user-defined operator implementations to be specified for operations where one or both of the operands are of a class or struct type (e.g. "Public static Complex operator+ (Complex c1, Complex c2)")
- ❖ Partial classes – allows you to split the definition of a class, interface, struct, method over two or more source files. Each file will contain a section of the definition. And all parts will be combined when the application is compiled
- ❖ Abstract classes – contain abstract methods which are implemented by derived class
    - Rules:
        - Cannot create an instance of an abstract class
        - Cannot declare an abstract method outside an abstract class
        - When a class is declared sealed, it cannot be inherited, abstract classes cannot be declared sealed
- ❖ Static member – no matter how many objects of the class are created, there is only one copy of the static member (can be variables or functions)
- ❖ Collections – specialized classes for data storage and retrieval
    - Serve various purposes such as allocating memory dynamically to elements and accessing a list of items on the basis of an index, etc. these classes can create collections of objects of Object class, which is base class for all data types in C#
        - ArrayList – represents ordered collection of an object, an alternative to an array. Can add and remove items from a list at a specified position using an index. Dynamic in size
        - Hash table – uses a key to access the elements in the collection. Each item in hash table is a key/value pair
        - SortedList – uses a key as well as an index to access the items in a list. A combination of an array and a hash table
        - Stack – last in first out (LIFO)
        - Queue – first in first out (FIFO)
        - BitArray – an array of the binary representation using 1 and 0
- ❖ Generics (T) – allow you to delay the specification of the data type of programming elements in a class or method. Until it is actually used in the program
    - Features:
        - Helps you maximize code reuse, type safety and performance
        - Can create your own generic interfaces, classes, methods, events and delegates
        - May create generic classes constrained to enable access to methods on particular data types
        - May get information on the types used in a generic data type at runtime by means of reflection

- Array[] vs ArrayList vs List<T>
  - Arrays –strongly typed and work well as parameters. If you know the length of your collection and it is fixed, you should use an array
  - ArrayLists – not strongly typed, every insertion or retrieval will need a cast to get back to your original type. If you need a method to take a list of a specific type, ArrayLists fall short because you could pass in an ArrayList containing any type. ArrayLists use a dynamically expanding array internally, so there is also a hit to expand the size of the internal array when it hits its capacity
  - List<T> - has all the advantages of Array and ArrayLists. It is strongly typed and it supports a variable length of items
- Anonymous Methods – provide a technique to pass a code block as a delegate parameter. Anonymous methods are methods without name, just body (don't need to specify return type, inferred from return statement in method body)
- "New" method modifier – hides a member that is inherited from a base class
- Predicate – function that returns true or false (i => i > 2) with lambda
  - Predicate delegate is a reference to a particular predicate useful for filtering lists of values
- Difference between Clone and Copy

Clone	Copy
Returns reference of an object in memory (replicate existing instance)	Copy the structure and data of the object (new instance)
Shallow copy	Deep copy

- Singleton – class which only allows one instance of itself to be created
  - Private constructor with no parameters (prevents classes from instantiating it)
  - Public static variable holds a reference to single created instance
- String vs StringBuilder

String	StringBuilder
Immutable instance	Mutable instance
Performance degrades with continuous changes	Shows better performance since new changes are made to existing instance
Better performance if string is not modified or rarely modified	Initializing of StringBuilder degrades performance (should not be used unless you have at least 4 string concatenations or other string manipulation happening)

ASP.NET Model-View-Controller (MVC)

- Model-View-Controller (MVC) is an architectural pattern which separates representation and user interaction
  - Model - represents the real world objects and provides data to the View
    - Responsible for how data should be handled
  - View – responsible for the look and feel of the application
    - Encapsulates presentation logic
    - Should not contain any application logic or database retrieval code

    - Renders appropriate UI by using data passed to it from Controller
- Controller – responsible for taking the end user request and loading the appropriate Model and View
    - Provides means for user output by presenting user commands and data
    - Receives output, translates it into appropriate messages and pass these messages to Views
    - Used for handling business logic.
    - Locating appropriate action method to call and validating that it can be called
    - Getting values to use as action method's arguments
    - Handles all errors that might occur during execution of action method
    - All controller classes must have "Controller" in the suffix name

❖ MVC Features
- Ideal for developing complex but light weight applications
- It is a Full Stack framework (has all tools required to build a complete website)
- It provides an extensible and pluggable framework which can be easily replaced and customized (can use your own view engines)
- Separation of concerns - Utilizes component based design of the application by logically diving it into Model, View and Controller components. Easy to manage complex large-scale projects and work on individual components
- Enhances Test Driven development and testability of application since all components can be designed interface-based and tested using mock objects
- Supports all existing ASP.NET functionalities such as Authorization, Authentication, Master Pages, Data Binding, user Controls, Routing, etc.
- Does not use concept of View State (present in ASP.NET) which makes application light-weight and gives full control to developers

❖ MVC Flow Diagram
- Client browser sends request to MVC application
- Global.asax receives request and performs routing (based on URL) to Controller
- Controller processes request and forms data model
- Model is passed to the appropriate View
- View renders the output

❖ Built in View engines
- Razor Engine (@) – markup syntax that enables the server side C#/VB code into web pages. Can be used to create dynamic content when page is being loaded. Razor is an advanced engine compared to ASPX engine
- ASPX Engine (<%-%>) – ASPX or Web Form engine is the default view engine that is included in the MVC framework since the beginning

❖ Web Forms vs MVC

Comparison Factors	Web Forms	MVC
Rendering Approach	Follows Page Control pattern approach for rendering layout	Front Controller approach is used
Separation of Concern	No separation of concerns and all Web Forms are tightly coupled	Very clean separation of concerns

Automated Testing	Automated testing is really difficult	TDD is quite simple in MVC because we can test separately
State	ViewState is used	Stateless
Performance	Slow due to large ViewState	Fast due to clean approach and no ViewState
Life Cycle	Web Form model follows a page life cycle	No page life cycle like Web Forms
Response time	Conversion logic which converts server controls to HTML output. Gets worse the more table/grid controls used	Twice as fast as Web Forms. No conversion
Reusability of code	Hard to reuse because code-behind is tightly coupled with forms	Easy to reuse
Controls	Lots of Server Side controls	No out of box controls (3rd party controls can be used)
Control over layout	Above abstract was good but limited controls over HTML, JS, CSS which is necessary in many cases	Full control over HTML, JS, CSS
RAD support	Yes	No
Scalability	Good for small scale application with limited team size	Better and recommended approach for large-scale applications

- ActionResult- Action Methods are responsible to execute request and generate response to it. By default, it generates response in form of ActionResult
- Action Filter – applies pre or post processing logic to a controller action and its result
  - Component we want to use to apply cross cutting logic to your application. Logic that we must execute across multiple controller actions, but do not want to duplicate logic inside of individual controllers
    - OutputCache – cache the output of an action method
    - Authorize – restrict an action or controller to authorize user or role
    - ValidateInput – to allow or disallow submission of html tags and other potentially dangerous content
    - ValidateAntiForgeryToken – helps prevent cross site request forgeries
    - HandleError – can specify a view to render in the event of an unhandled exception
- Routing – used to hide data which is not intended to be shown to final user
  - Handles MVC's URLs. Let's you create any URL pattern you desire and express them in a clear and concise manner
  - Each route contains a specific URL pattern. This pattern is compared to the incoming request URLs and if it matches the pattern, it is used by routing engine to further process the request
- Strongly Typed View – view is tightly coupled to Model
- Loosely Typed View – view is not tightly coupled to Model and same view can use any Model

- ❖ Partial Views – reusable view which can be embedded inside other views (like user control in WebForms)
  - Use RenderPartial() to call a partial view in main view
- ❖ Layout – common site template (like Masterpage in WebForms)
  - Use RenderBody() to place dynamic content in view
- ❖ Html Helpers – methods that return a string. Can represent any type of content you want. Can be used to render standard tags like <input> or more complex content such as <table> of DB data
  - Standard Html Helpers: ActionLink, BeginForm, CheckBox, DropDownList, ListBox, Password, RadioButton, TextArea, TextBox, Hidden
- ❖ ViewData – dictionary object which is used to pass data from controller to view. Its life only lies during current request. If redirection occurs, then its value becomes null. Required typecasting for getting data and check null values to avoid error
- ❖ ViewBag – dynamic property used to pass data from controller to view. Life lies only during current request. If redirection occurs its value becomes null. Doesn't require typecasting for getting data
- ❖ TempData – dictionary object used to pass data from current request to subsequent requests (redirecting from one page to another) life lies only till target view is fully loaded. Required typecasting to get data and check null values to avoid error. Used to store only one time messages like error messages, validation messages
- ❖ Model Binders - what Model binder does is that it takes the values coming from an HTML page and map it to a corresponding model. This is a very useful since it relieves the developers from writing all the "type casting" and "HTML-Model mapping" code
- ❖ Data Annotations and Validation – are attribute classes in the namespace
  - Required – Indicates that the property is a required field
  - DisplayName – Defines the text we want used on form fields and validation messages
  - StringLength – Defines a maximum length for a string field
  - Range – Gives a maximum and minimum value for a numeric field
  - Bind – Lists fields to exclude or include when binding parameter or form values to model properties
  - ScaffoldColumn – Allows hiding fields from editor forms
- ❖ MVC Versions
  - 6.0:
    - ASP.NET MVC and Web API has been merged in to one.
    - Dependency injection is inbuilt and part of MVC.
    - Side by side - deploy the runtime and framework with your application
    - Everything packaged with NuGet, Including the .NET runtime itself.
    - New JSON based project structure.
    - No need to recompile for every change. Just hit save and refresh the browser.
    - Compilation done with the new Roslyn real-time compiler.
  - 5.0:
    - Attribute based routing
    - Asp.Net Identity
    - Bootstrap in the MVC template

    - Authentication Filters
    - Filter overrides

- 4.0:

    - ASP.NET Web API
    - Refreshed and modernized default project templates
    - New mobile project template
    - Many new features to support mobile apps
    - Enhanced support for asynchronous methods

- 3.0:

    - Razor
    - Readymade project templates
    - HTML 5 enabled templates
    - Support for Multiple View Engines
    - JavaScript and Ajax
    - Model Validation Improvements

SQL Server Integration Services (SSIS)

❖ SQL server integration services (SSIS) introduces the concepts of control flow and data flow in packages
- Tool that we use to perform Extract, Transform, Load (ETL) operations.
- Can create a Maintenance plan using SQL Server Management Studio (SSMS) an SSIS package
- Retrieve data from just about any source
- Perform various transformations on the data (e.g. Convert from one type to another, lowercase/uppercase, perform calculations, etc.)
- Load data into just about any source
- Define a workflow

❖ SSIS Package – discrete, named collections of connections, control flow and data flows that implement data movement/transformation
- Control flow and tasks – one or more tasks and container drive what the package does. You organize control flow based on what you want package to do. Tasks are actions taken to accomplish the desired data movement/transformation. Task can execute any SQL statement, send mail, bulk insert data, etc.
- Containers – groups one or more related tasks that you want to manage together (reuse)
- Workflows – definable precedence constraints that allow you to link two tasks, based on whether the first task executes, executes successfully, or unsuccessfully

  Three types of precedence constraints:

    - Unconditional – does not matter whether the preceding step failed or succeeded
    - On success – preceding step must have been successful for execution of next step

    - On failure – returns the appropriate error
- Data flow – identifies the sources and destinations that extract and load data. Identifies transformations that manipulate or enhance the data. Provides paths that link sources, transformations and destinations
- Data flow task – creates, orders and runs the data flows themselves, using data flow engine
- Transformations – one or more functions or operations applied against a piece of data before the data arrives at the destination
- Event handlers – these workflow tasks run in response to events raised by package, task or container

SQL Server Reporting Services (SSRS)

❖ SQL server reporting services (SSRS) is a comprehensive reporting platform whereby reports are stored on a centralized inver. Because reports are centralized, users run reports from one place (report deployment is quite simplified)
- Provides several extensions towards data rendering, delivery, and security reports
- Reports can be mobile, paginated, interactive, tabular and graphical
- Variety of data – relational, multidimensional, XML-based data sources
- Rich data visualizations – charts, maps, sparklines and key performance indicator (KPI)

Web Services (WS) ASMX

❖ Web services are web application components. Can be published, found and used on the web
- WS are application components
- Communicate using open protocols
- Self-contained and self-describing
- Can be discovered using UDDI (Universal Definition, Discovery and Integration)
- Can be used by other applications
- HTTP and XML is basis for WS
    - By using WS your app can publish functions or messages to the rest of world
    - Uses XML to code and decode data
    - Uses SOAP to transport it using open protocols

❖ Two types of WS –
- Reusable application components
    - Things applications need often do not need to be make them again (currency conversion, weather reports, language translation, etc.)
- Connect existing software
    - Can help solve the interoperability problem by giving different applications a way to link their data
    - Can exchange data between different applications and platforms
    - Any application can have a WS component
    - Can be created regardless of programming language

❖ Simple Object Access Protocol (SOAP)

- Communication protocol
- Format for sending and receiving messages
- Platform independent
- Based on XML

❖ SOAP Building Block -Ordinary XML document containing:
- ENVELOPE element that identifies XML document as SOAP message (required root element)
- HEADER element contains header information (optional. Authentication, payment, etc.)
- BODY element contains call and response information (required)
- FAULT element contains errors and status information (optional, child of BODY)

Syntax Rules:

- Must be enclosed using XML
- must use SOAP Envelope and Encoding
- must not contain DTD reference and XML processing instructions

❖ Web Services Description Language (WSDL) - describes a web service. Specifies location and methods of service using:
- <types>defines data types used by WS
- <message> defines data elements for each operation
- <portType> describes operation that ca be performed and messages involved
- <binding> defines protocol and data format for each port type

4 WSDL Request-Response types:

- One-way – operation receive request and no return
- Request-response – operation receive request and return response
- Solicit-response – operation send request and wait for response
- Notification – operation send message and will not wait for response

❖ Universal Description, Discovery and Integration (UDDI) – platform independent framework for describing services, discovering businesses and integrating business services over internet
- Directory for storing information about WS
- Directory of WS interfaces described by WSDL
- Communicates via SOAP
- Built into .NET platform

Benefits:

- Makes it possible to discover right business from millions online
- Defines how to enable commerce once preferred business is discovered
- Reaching new customers and increasing access to current
- Expanding offerings and extending market reach
- Solving customer driven need. Allow for rapid participation in global internet economy
- Describes services and business processes in a single, open and secure environment

❖ Extensible Markup Language (XML)
- Used to store and transport data
- Self-descriptive

- No predefined tags
- Hierarchical
- Can be passed and used by a lot of programming languages
- Can be fetched with XMLHttpRequest

XML Schema – describes the structure of an XML document

- Well-Formed is an XML with correct syntax

Simplifies:

- Data sharing – compatible format with all systems
- Data transport – stores data in plain text format. Software and hardware independent
- Platform changes – easy to expand or upgrade to new systems without losing data
- Data availability – to all reading machines

❖ JavaScript Object Notation (JSON)
- Lightweight data-interchange format
- Language independent
- Self-describing and easy to understand
- Can use standard JS functions to convert JSON to native JS Object

❖ JSON vs XML
- JSON doesn't need end tags
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays
- XML has to be parsed

❖ Serialization and Deserialization
- Serialization – process of converting an object into a stream of bytes
- Deserialization – extracting data structure of a stream of bytes

Windows Communication Foundation (WCF)

❖ WCF is a framework for building service-oriented applications. Can send asynchronous messages from one service endpoint to another
- Supports HTTP, TCP, Named-Pipe and MSMQ
- Supports SOAP, REST, POX
- Can be hosted on different types of application (WS can only be hosted on web server)
- Built in support for transaction and reliable sessions

❖ Basic Requirements of WCF:
- Server
  - Define and implement a Service contract
  - Construct a Service Host Instance for service type exposing endpoints
  - Open communication channel
- Client
  - Requires a copy of service contract and information about endpoints

- Construct a communication channel for particular endpoint and call operations

❖ Service Endpoints
- Used by Service Host to receive the request from the client and forward to service
- Service can have one or more endpoints
- Every endpoint will have unique combination of ABC
- Can be defined in code or configuration file (config file allows you to change endpoints when needed. No one can change endpoints if code is built and deployed)

❖ ABC
- Address – where to send
- Binding – how to send
- Contract – what message looks like

❖ Endpoints in configuration file
- Everything will be under <system.serviceModel>
- <services> contains service type definitions
- <endpoints> contains one or more binding sections to configure individual bindings
- <behavior> provides configuration for services

❖ Bindings – attribute of an endpoint and it lets you configure channels like transport protocol, encoding and security requirements
- Types of Bindings: BasicHttpBinding, WSHttpBinding, WSDualHttpBinding, WSFederationHttpBinding, NetNamePipeBinding, NetTcpBinding, NetPeerTcpBinding, NetMsmqBinding, MsmqIntegrationBinding

❖ ServiceContract
- maps a CLR interface to a technology-neutral service contract
- It describes what the service can do
- Defines some properties about the service, and a set of actions called Operation Contracts.

Properties:

- ConfigurationName – specifies the name of service element in config file to use
- Name/Namespace – control the name and namespace of the contract in the WSDL <portType> element
- SessionMode – specifies whether the contract requires a binding that supports sessions
- CallbackContract – specifies the return contract in a two-way conversation
- HasProtectionLevel/ProtectionLevel – indicate whether all messages supporting contract have an explicit ProtectionLevel value, and if so what level it is

❖ Data Contract
- Describes how CLR types are mapped to XSD schema definitions
- Enables complex types to be serialized and deserialized so it can be exchanged between client and server (DataContractSerializer)
- Loosely coupled system. As long as datapassed between services, conforms to same contract, all services can process data

Properties:

- Name – gets or sets the name of data contract for type

- Namespace – gets or sets the namespace for the data contract for type

❖ Data Member Attribute
- Applied in conjunction to DataContractAttribute to identify members of the type that are part of DataContract
- Can apply DataMemberAttribute to private fields or properties. Data returned by member will be serialized and deserialized and thus can be viewed or intercepted by malicious user or process
- Properties with DataMemberAttribute must have both get and set fields

Properties:

- IsRequired – gets or sets a value that instructs serialization engine that member must be present when reading or deserializing
- Name – gets or sets a data member a name
- Order – gets or sets the order of serialization and deserialization of a member

❖ Handling WCF Exceptions / Faults Contract
- WCF does not communicate CLR Exceptions
- Handle error situations by mapping managed exception objects to SOAP fault objects to be passed to client by using FaultContract
- Consists of SOAP Fault messages
- For security purpose, exception message doesn't contain information about actual exception (can be configured to return more detailed exception information)

❖ Message Exchange Patterns – 3 types
- Request - Reply – client sends message to service and waits for service response
- One way – client sends message and doesn't wait for response (used for logging and storing data. IsOneWay property)
- Duplex – client and service initiate communication by sending message to each other (use for service to notify client it finished processing operation or when)

❖ Transaction – Logical units of work comprising of multiple activities that should all succeed or all fail
- ACID
  - Atomic – transaction executes at once and all of the work occurs or none does
  - Consistency – needs to preserver data consistency (commit or rollback)
  - Isolation – needs to be independent of other transactions
  - Durable – transactions are recoverable. Needs to log its state so it should know what to undo if fails

❖ Microsoft Message Queuing (MSMQ) – technology for asynchronous messaging
- Can be used whenever there's need for two or more applications to send messages to each other without needing immediate results
- Can communicate with other machines over internet
- Similar to an e-mail server but designed to let applications talk to each other

❖ Security

Concepts:

- Authentication – client and service have clear identity. Client has to provide login credentials
- Authorization – client has permissions or not to call operations of service

- Confidentiality only client can read data passed between client and service. Data is encrypted on network
- Integrity – messages have not been tampered with when they are on network

Mechanism:

- Transport Level Security – applied at OS level at network or transport level and before message is sent

  Encryption based on binding

  - HTTPS uses SSL
  - TCP uses Transport Layer Security (TLS)
  - Can require clients to pass credentials for authentication

  Pros: faster and can benefit from hardware acceleration

  Cons: provides only point to point encryption. Intermediary service can decrypt and alter it

- Message Level Security
  - Messages are encrypted based on WS-Security standard and signed before they are sent
  - Can require client to pass credentials for authentication

  Pros: provides end to end encryption since message is encrypted

  Cons: slower. Required both client and service to support WS-Security

❖ REST – used to expose a public API over the internet to handle CRUD operations on DATA focused on accessing named resources through a single consistent interface
- Uses standard HTTP much simpler in every way
- Permits many different data formats
- Better performance and scalability (reads can be cached)

❖ SOAP – brings its own protocol and focuses on exposing pieces of application logic (operations, not data) as services. Focused on accessing named operations, each implement some business logic through different interfaces
- WS-Security adds some enterprise security features (not just point to point SSL)
- WS-AtomicTransaction – ACID transactions over services (REST transactions are not ACID)
- WS-ReliableMessaging – successful/retry logic built in and provides end to end reliability

❖ REST vs SOAP

REST	SOAP
Easier to use and more flexible	More heavyweight and harder to use
Efficient (can use different data formats like JSON which is lighter than XML)	Standardized (XML only)
Fast (no extensive processing required)	Provides pre-built WS-Standards
No expensive tools required to interact with web service. Uses HTTP	Language, platform and transport independent (rest requires HTTP)
Small learning curve	Larger learning curve
No built-in error handling	Has built-in error handling

Assumes point to point communication	Has enterprise security, more secure. Works well with distributed enterprise environments

Web API

- Web API is a framework for building and consuming HTTP services that can reach a broad range of clients including browsers, phones, and tablets
  - Ideal platform for RESTful applications on .NET Framework
  - Can use either XML or JSON as output. (JSON is good for mobile)
  - Takes best features from WCF and merges them with best features from MVC. (WCF needs a lot of configurations)
- Why Web API
  - If we do not need SOAP
  - If we want to build a simple non-SOAP-based HTTP services on top of existing WCF message pipeline
  - We don't need to do extensive configuration like WCF
  - Light weight architecture and good for devices which have limited bandwidth like mobile phones
  - Open source
- Routing – When request is received it routes to the action in controller
- Controller – Class that handles HTTP requests
- Methods – Get, Put, Post, Delete

JavaScript and JQuery

- Software Library – Collection of implementations of behavior with well-defined interface by which behavior is invoked, reuse behavior, modularity (jQuery)
- Software Framework – Abstraction in which software provides generic functionality that can be selectively changed by additional user-written code, universal reusable environment that provides particular functionality as part of a larger software platform (AngularJS, Ember, Backbone)
- Library vs Framework
  - Library – collection of functions which are useful when writing when apps. Your code is in charge and it calls into the library when it sees fit
  - Frameworks – a particular implementation of a web application, where your code fills in the details. Framework is in charge and it calls into your code when it nee1111111111ds something app specific
    - Full-Stack – idea of having everything you need to build a web app in one framework
    - Part-Stack – requires an accompanying framework to handle other sections of application
- JavaScript Frameworks
  - Single Page application (SPA)
    - Web apps that load a single HTML page and dynamically update that page (no page refresh on change)
    - Rich internet applications
    - Client side frameworks are the way to build rich web apps (AngularJS, Backbone, Batman, Ember...)

- Model View Controller (MVC)
  - All technologies made use of model view separation (MVC, MVVM, MV+)
  - Data binding, routing
- Scalable, reusable, maintainable JS code
- Test Driven Development (TDD)

AngularJS

❖ AngularJS is a structural framework for dynamic web apps (it is part-stack framework)
- HTML only does static documents. Angular fills in the gap to support dynamic application
  - Solving the impedance mismatch
  - Designed with CRUD application (data-driven) in mind
  - Declarative approach

❖ Why AngularJS
- Single Page application
- Powerful JavaScript based development framework to create RICH Internet Application (RIA)
- AngularJS provides options to write client side application (using JS) in a clean MVC way
- Application written in Angular is cross-browser compliant. Angular automatically handles JS code suitable for each browser
- Open source, supported by Google, large community, a lot of resources
- One of the most popular front-end JS frameworks

❖ Core Features
- Data-binding – automatic synchronization of data between model and view components
- Scope – objects that refer to the model. They act as glue between controller and view. Controllers and directives have reference to scope. But no each other. Isolates controller from directive and DOM
- Controller – JavaScript attributes/properties and functions that are bound to a particular scope. Exposes variables and functionality to expression and directives
- Services –These are singleton objects which are initiated only once in app. provides separation of concern, more manageable, testable, and reusable code. Built-in services $http, $resource, $route, $scope, $rootScope, $parse, $templatecache, $animate, $location, etc. (30 of them). Can create our own services and use own services inside filters
- Filters – these select a subset of items from an array and returns a new array (do not modify the underlying data) can be used in view templates, controllers or services. Angular has some built-in filters like “orderBy” or create a custom filter (pipe character “|” to define filters). Some filters (LowerCase, UpperCase, LimitTo, Date, Currency, Json, Number)
- Directives – directives are markers on DOM elements (such as elements, attributes, css) these can be used to create custom HTML tags that serve as new custom widgets. Angular has built in directives (ng-app, ng-controller, ng-bind, ng-model, ng-init, ng-repeat, etc.)
- Templates – rendered view with information from the controller and model. Can be single file (like index.html) or multiple views in one page using “Partial” (written with HTML and contains Angular elements and attributes)

- Routing – switching views and links controllers to views
- Model View Whatever (MV*) or Model View ViewModel (MVVM) – statefulness added to MVC. Data is cached on the client (and along with it, intelligence to handle that data) called ViewModel. While MVC is essentially a one-way communication (poor interactivity), MVVM is a two-way communication (rich interactivity)
- Modules – collection of controllers, directives, filters, services, and other configuration information
- Deep Linking – allows you to encode the state of application in URL so it can be bookmarked. Can be restored from URL to same state
- Dependency Injection – software design pattern that implements inversion of control for resolving dependencies. A dependency is an object that can be used (service). An injection is the passing of a dependency to a dependent object (client) that uses it. Angular has built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand and test

❖ Advantages
- Provides capability to create Single Page application in a clean and maintainable way
- Provides data binding capability to HTML for a rich and responsive experience
- Uses dependency injection and make use of separation of concerns
- Code is unit testable
- Provides reusable components
- Developers write less code and get more functionality
- Can run on all major browsers and smart phones

❖ Disadvantages
- Not secure – being JS only framework. Server side authentication and authorization is a must to keep an application secure
- Not degradable – if user disables JS then user will just see the basic page

❖ Angular Directives
- Ng-app – defines and links an Angular to HTML
- Ng-init – evaluate an expression and initialize JS variable
- Ng-model – binds the values of Angular scope data to HTML input controls (vice-versa) (input)
- Ng-bind – binds the Angular data to HTML tags (output)
- Ng-repeat – loops over items in a collection, instantiates a template for each item
- Ng-Route – routing helps divide application in logical views and bind different views to controllers

❖ Angular Expressions
- Evaluated against an Angular scope object
- No conditionals, loops, exceptions
- Expressions enclosed in {{expression}}

❖ Two-way Data binding – binding an HTML or CSS property to JS variable. When the value of variable is updated, HTML/CSS property is also updated

❖ Angular Forms – collection of controls for the purpose of grouping related controls together
- Provides Angular validation services (server side validation is still necessary for a secure application)
    - Required fields

- Input format pattern (email, phone, number, url, etc.)
- Max and Min value
- User gets instant feedback

- AngularJS provides directives for some of the HTML5 input types to handle validation (for input types not which are not provided by Angular, use ng-pattern attribute to handle)

States:

- Input field states - $untouched, $touched, $pristine, $dirty, $invalid, $valid
- Form states - $submitted, $pristine, $dirty, $invalid, $valid

❖ Angular Services
- $http – facilitates communication with the remote HTTP servers. Used to generate an HTTP request and returns a promise (.config, .data, .headers, .status, .statusText)
- $resource – is a factory which creates a resource object that lets you interact with RESTful server side data sources
- $location – has methods that return information about the location of current web page
- $rootScope – every application has a root Scope. All other scopes are descendent scopes of the root. There can only be one and is shared among all components of an app (acts like a global variable)

❖ Angular UI-Router – routing framework for AngularJS
- Provides a different approach than ngRoute in that it changes your application views based on state of application, not just route URL

❖ Angular Testing – Tools used for testing: Karma, Jasmine, Angular-mocks

❖ Exception Handling – any uncaught exception in angular expressions is delegated to this service $exceptionHandler. Or we can use catch/throw manually

❖ Events directives – allows us to run Angular functions at certain user events (angular event does not override HTML event; both will be executed)
- Ng-blur, ng-change, ng-click, ng-copy, ng-cut, ng-dblclick, ng-focus, ng-keydown, ng-keypress, ng-keyup, ng-mousedown, ng-mouseenter, ng-mouseleave, ng-mousemove, ng-mouseup, ng-paste
- Use ".on" function to handle events (element.on("click", function(){ };)

❖ Animation – provides animated transitions, with help from CSS
- Must include angularjs animate library and refer ngAnimate module in application or ngAnimate use dependency injection
- ngAnimate does not animate HTML elements. It gives you some predefined classes which are used to make animations (ng-show, ng-hide, ng-class, ng-view, ng-include, ng-repeat, ng-if, ng-switch)
- Use CSS to add effect to animation

❖ AngularJS SQL – perfect for displaying data from a database (data must be in JSON format)
- Cross-Site HTTP requests – requests for data from different server (than the requesting page)
- Used to load css, images and scripts from different servers
- Most browsers restrict cross-site requests from scripts to same site for security reasons. You need to add access to allow it ("Access-Control-Allow-Origin")

❖ AngularJS Global API (Application Programming Interface) – is a set of global JavaScript functions for performing common tasks like comparing objects, iterating objects, converting data (accessed using angular object)

- Angular.lowercase(), .uppercase, .isString(), .isNumber()

Bootstrap

- Boostrap is the most popular HTML, CSS, JavaScript framework for developing response, mobile-first websites
  - Open source front-end framework for faster and easier web development
  - Includes HTML and CSS design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
  - Gives you the ability to easily create responsive design (automatically adjusts look on all size devices)

  Advatanges

    - East to use – anybody with basic html and css knowledge can use it
    - Responsive features – adjusts to phones, desktop, etc
    - Mobile-first approach – mobile first styles are part of the core framework
    - Browser compatibility – compatible with all modern browsers
- Grid system – used for creating page layouts through a series of rows and columns
  - allows up to 12 columns across the page
  - Appropriately scales up to 12 columns depending on device size
  - Includes predefined classes for easy layout options
  - Can group them together however you want (.col-md-6)
- Jumbotron (.jumbotron)– indicates a big box for calling extra attention to some special content or information (display as a grey box with rounded corners, enlarges with content inside)
- Typography – default font-size is 14px with line-height of 1.428. applied to <body> and paragraphs
- Tables (.table)– basic table has light padding and only horizontal dividers
- Page Header (.page-header) – like a section divider. Adds a horizontal line under the heading
- Wells (.well) – add a rounded border around an element with gray background color and some padding
- Alerts (.alert) – provides easy way to create predefined alert messages
- .alert-success (green), -info (blue), -warning (yellow), -danger (red)
- Buttons (.btn) - .btn-default (white), -primary (darkblue), -success (green), -info (lightblue), -warning (yellow), -danger (red), -link
- Button Groups (.btn-group) – allows you to group a series of buttons together (on a single line) in button group
- Glyphicons (.glyphicon) – provides 260 glyphicons
- Badges (.badge) – numerical indicators of how many items are associated with a link. Use within a span class
- Progress Bar (.progress) – can be used to show a user how far along he/she is in process
- Pagination (.pagination) – to list of links
- Pager (.pager) – for buttons like Previous and Next
- List Groups – (.list-group) for <ul> and (.list-group-item) for
- Panels (.panel) – bordered box with some padding around its content

- ❖ Basic Dropdowns (.dropdown) – adds styling to toggleable menu that allows users to choose one value from list
- ❖ Collapse (.collapse) – useful when you want to hide and show large amount of content

Internet Information Services (IIS)

- ❖ Internet information services (IIS) is an extensible web server created by Microsoft for use with Windows
  - Supports HTTP, HTTPS, FTP, FTPS, SMTP, NNTP

NUnit

- ❖ NUnit is an open source unit testing framework for Microsoft .NET
  - Tests can run from a console runner in VS through Test Adapter
  - Tests can be run in parallel
  - Strong support for data driven tests
  - Every test case can be added to one or more categories, to allow selective running
  - Supports multiple platforms (.NET Core, Xamarin Mobile, Compact Framework, Silverlight)
- ❖ Console Runner (nunit3-console.exe) – used for batch execution of tests. Works through the NUnit Test Engine, which provides the ability to load, explore and execute tests. When they are to be run in separate process, the engine makes use of the nunit-agent program to run them
  - NUnitLite runner may be used in situations where a simpler runner is more suitable. It allows developers to create self-executing tests
- ❖ NUnit provides a rich set of assertions as static methods of the Assert class. If an assertion fails, the method call does not return and an error is reported. If a test contains multiple assertions, any that follow the one that failed will not be executed. For this reason, it's usually best to try for one assertion per test
- ❖ bah being all nervous about that interview
- ❖ don't even think it lasted 10 minutes
- ❖ he asked me literally 5 questions Abhi
- ❖ it wasn't that serious
- ❖ Asked me to talk about myself a little, trick question of if I'd do validation on client or server side (said both), order of a page view IE what happens when you call a page, and how to secure web services