**Net Interview Question Bank**

**Fresher Academy**

**Updated: 16/08/2017**

# C#

1. What is Class?
   * + - A class is like a blueprint of specific object
       - In C#, a class can be defined by using the class keyword
       - **C# Class** defines properties, fields, events, methods
       - Access modifiers defines the accessibility of a class
       - **Namespace** can include one or more classes
2. C# variable
   * + - The variable is a name given to a data value
       - A variable holds the value of specific data type
       - A variable can be declared and initialized later or at the same time.
       - The value of a variable can be changed at any time
       - Multiple variables can be defined separated by comma (,)
       - A value must be assigned to a variable before using it
3. C# Data Types
   * + - The data type tells a C# compiler what kind of value a variable can hold
       - C# includes many in-built data types for different kinds of data
       - Each data types includes specific range of values
4. Value Type and Reference Type
   * + - Value type stores the value in its memory space
       - Reference type stores the address of the value where it is stored.
       - Primitive data types and struct are of the 'Value' type
       - Class objects, string, array, delegates are reference types.
       - Value type passes byval by default. Reference type passes byref by default
       - Value types and reference types stored in Stack and Heap
5. C# Keywords
   * + - Keywords are reserved words that cannot be used as name or identifier.
       - Prefix '@' with keywords if you want to use it as identifier.
       - C# includes various categories of keywords
       - Contextual keywords can be used as identifier
6. C# Interface
   * + - An Interface only contains declarations of method, events & properties.
       - An Interface can be implement implicitly or explicitly.
       - An Interface cannot include private members
       - All the members are public by default
7. C# Operators
   * + - Operator in C# is a special symbol that specifies
       - operators have different meanings based on the datatype of the operand
8. If statement
   * + - If-else statement controls the flow of program
       - It should start from the If statement
       - Only one else statement is allowed in the if-else chain.
       - Multiple else-if statements are allowed in a single if-else chain.
       - Nested if-else statement is allowed
9. C# switch
   * + - The switch statement tests the variable against a set of constants.
       - The switch statement contains multiple case labels.
       - The switch case includes break keyword to stop the execution of switch case.
       - The default case executes when no case satisfies the expression.
       - A nested switch statement is allowed
10. C# for loop
    * + - The for loop executes the block of code repeatedly.
        - The for loop has three steps: initialization, condition and increment/decrement.
        - The for loop can use control variable of any numeric data type.
        - Use break keyword to stop the execution and exit from for loop.
        - Nested for loop is allowed in C#
11. C# while loop
    * + - The while loop executes the block of code repeatedly.
        - The while loop includes condition expression
        - Use break keyword to stop the execution and exit from while loop.
        - An nested while loop is allowed
12. C# do-while
    * + - The do-while loop executes the block of code repeatedly
        - The do-while loop execute the code at least once
        - Use the break keyword to stop the execution and exit from a do-while loop.
        - An nested do-while loop is allowed
13. Structure in C#
    * + - Structure is a value type and defined using struct keyword.
        - Structure can be initialized with or without new keyword.
        - Structure must be initialized with new keyword
14. Enum in C#
    * + - The enum is a set of named constant.
        - The value of enum constants starts from 0
        - Enum can have value of any valid numeric type.
        - String enum is not supported in C#.
        - Use of enum makes code more readable and manageable
15. C# StringBuilder
    * + - StringBuilder is mutable.
        - StringBuilder performs faster than string when appending multiple string values
        - Initialize StringBuilder as class
        - Use StringBuilder when you need to append more than three or four strings.
        - Use Append() method to add or append strings with StringBuilder
        - Use ToString() method to get the string from StringBuilder
16. C# Array
    * + - An Array stores values in a series starting with a zero-based index.
        - The size of an array must be specified while initialization.
        - An Array values can be accessed using indexer.
        - An Array can be single dimensional, multi-dimensional and jagged array.
        - The Array helper class includes utility methods for arrays
17. Multi-dimensional Array
    * + - A multi-dimensional array is a two dimensional series like rows and columns
        - The values of a multi-dimensional array can be accessed using two indexes
        - The first index is for the row and the second index is for the column
        - Both the indexes start from zero
18. C# Collection
    * + - C# collection holds many values or objects in a specific series
        - C# collections have two types: non-generic collections and [generic collections](http://www.tutorialsteacher.com/csharp/csharp-generic-collections)
        - Every collection class implements the [**IEnumerable**](http://msdn.microsoft.com/en-us/library/system.collections.ienumerable(v=vs.110).aspx) interface
19. C# ArrayList
    * + - ArrayList can store items (elements) of any datatype.
        - ArrayList resizes automatically as you add the elements.
        - ArrayList values must be cast to appropriate data types before using it.
        - ArrayList can contain multiple null and dulplicate items.
        - ArrayList can be accessed using foreach or for loop or indexer.
        - Use Add(), AddRange(), Remove(), RemoveRange(), Insert(), InsertRange(), Sort(), Reverse() methods
20. C# SortedList
    * + - C# has generic and non-generic SortedList.
        - SortedList stores the key-value pairs in ascending order of the key. Key must be unique and cannot be null whereas value can be null or duplicate.
        - Non-generic SortedList stores keys and values of any data types. So values needs to be cast to appropriate data type.
        - Key-value pair can be cast to DictionaryEntry.
        - Access individual value using indexer. SortedList indexer accepts key to return value associated with it
21. C# Stack
    * + - Stack stores the values in LIFO (Last in First out) style
        - Use the Push() method to add elements into Stack.
        - The Pop() method returns and removes elements from the top of the Stack
        - Calling the Pop() method on the empty Stack will throw an exception
        - The Peek() method always returns top most element in the Stack
22. C# Queue
    * + - The Queue stores the values in FIFO (First in First out) style
        - Use the Enqueue() method to add elements into Queue
        - The Dequeue() method returns and removes elements from the beginning
        - Calling the Dequeue() method on an empty queue will throw an exception.
        - The Peek() method always returns top most element
23. C# Hashtable
    * + - Hashtable stores key-value pairs of any datatype where the Key must be unique.
        - The Hashtable key cannot be null whereas the value can be null.
        - Hashtable retrieves an item by comparing the hashcode of keys
        - Hashtable uses the default hashcode provider
        - Use DictionaryEntry with foreach statement to iterate Hashtable
24. C# Indexer
    * + - An indexer is same as property except
        - Indexer can be override by having different types of parameters.
        - Ref and out parameter with the indexer is not supported.
        - Indexer can be included as an interface member
        - Use code snippet to insert indexer syntax automatically in the visual studio
25. C# Stream
    * + - Stream is an abstract class for transferring bytes from different sources
        - FileStream class provides reading and writing functionality of bytes
        - Reader & writer classes provides functionality to read bytes and converts bytes
        - StreamReader provides a helper method to read string from FileStream
        - StreamWriter provides a helper method to write string to FileStream
26. Working with Files & Directories
    * + - File is a static class to read\write from physical file with less coding.
        - Static File class provides functionalities such as create, read\write, copy, move, delete and others for physical files.
        - Static Directory class provides functionalities such as create, copy, move, delete etc for physical directories with less coding.
        - FileInfo and DirectoryInfo class provides same functionality as static File and Directory class.
27. C# FileInfo
    * + - Use FileInfo class to perform operations on physical files manually.
        - Use FileInfo for reading or writing string from physical file
28. Exception in C#
    * + - Exception is a base class for any type of exception class in C#.
        - Exception types: SystemException and ApplicationException.
        - SystemException class is used for CLR related runtime errors.
        - Exception class includes important properties
29. Exception Handling in C#
    * + - Use the try, catch and finally blocks to handle exceptions in C#.
        - The try block must be followed by a catch or finally block or both.
        - A multiple catch block is allowed with different exception filters.
        - General catch{..} block must come last.
        - catch{..} and catch(Exception ex){ } both cannot be used.
        - The finally block must come after the try or catch block.
        - The finally block will always execute irrespective
        - The finally block is appropriate place for disposing objects.
        - The finally block cannot have a return or break
        - Nested try-catch blocks are allowed in C#.
        - An Exception will be caughtin the inner catch block if appropriate filter found
30. C# - throw keyword
    * + - An exception can be raised manually by using the throw keyword
        - The throw keyword only used to raise Exception
31. Create Custom Exception Class in C#
    * + - Exception is a base class for any type of exception class in C#.
        - Derive Exception class to create your own custom exception classes
32. C# Delegate
    * + - Delegate is a function pointer. It is reference type data type.
        - Syntax: public delegate void <function name>(<parameters>)
        - A method that is going to assign to delegate must have same signature as delegate.
        - Delegates can be invoke like a normal function or Invoke() method
        - Multiple methods can be assigned to the delegate using "+" operator
33. C# Event
    * + - Use event keyword with delegate type to declare an event.
        - Check event is null or not before raising an event.
        - Subscribe to events using "+=" operator. Unsubscribe it using "-=" operator.
        - Function that handles the event is called event handler
        - Events can have arguments which will be passed to handler function.
        - Events can also be declared static, virtual, sealed and abstract.
        - An Interface can include event as a member.
        - Events will not be raised if there is no subscriber
        - Event handlers are invoked synchronously if there are multiple subscribers
        - The .NET framework uses an [EventHandler](http://msdn.microsoft.com/en-us/library/system.eventhandler.aspx) delegate and an [EventArgs](http://msdn.microsoft.com/en-us/library/system.eventargs.aspx)base class.
34. Generics in C#
    * + - Generics denotes with angel bracket <>.
        - Compiler applies specified type for generics at compile time.
        - Generics can be applied to interface, abstract class, method, static method, property, event, delegate and operator.
        - Generics performs faster by not doing boxing & unboxing
35. Constraints in Generics
    * + - Constraints specifies the kind of types allowed with the generics.
        - Constraints can be applied using the where keyword.
        - Six types of constraints can be applied
        - Multiple constraints also can be applied.
36. Generic Collection in C#
    * + - A generic collection gets all the benefit of generics
        - It doesn't need to do boxing and unboxing
37. C# List<T>
    * + - List<T> stores elements of the specified type and it grows automatically.
        - List<T> can store multiple null and duplicate elements.
        - List<T> can be assigned to IList<T> or List<T> type of variable
        - List<T> can be access using indexer, for loop or foreach statement.
        - LINQ can be used to query List<T> collection.
        - List<T> is ideal for storing and retrieving large number of elements
38. C# Generic SortedList
    * + - C# has a generic and non-generic SortedList.
        - SortedList stores the key-value pairs in ascending order of the key. The key must be unique and cannot be null whereas value can be null or duplicate.
        - Generic SortedList stores keys and values of specified data types. So no need for casting.
        - Key-value pair can be cast to a KeyValuePair<TKey,TValue>.
        - An individual value can be accessed using an indexer. SortedList indexer accepts key to return value associated with it
39. C# Dictionary
    * + - A Dictionary stores Key-Value pairs where the key must be unique.
        - Before adding a KeyValuePair into a dictionary, check that the key does not exist
        - Use the TryGetValue() method to get the value of a key
        - Use a foreach or for loop to iterate a dictionary.
        - Use dictionary indexer to access individual item.
        - Use custom class that derives IEqualityComparer to compare object
40. C# Partial Class
    * + - Use the partial keyword to split interface, class, method or structure into multiple .cs files.
        - The partial method must be declared before implementation.
        - All the partial class, method , interface or structs must have the same access modifiers
41. C# Static
    * + - Static classes cannot be instantiated using the new keyword
        - Static items can only access other static items
        - Static items share the resources between multiple users
        - Static cannot be used with indexers, destructors or types other than classes
        - A static constructor in a non-static class runs only once
        - A static constructor in a static class runs only once
        - Static members are allocated in high frequency heap area of the memory
42. C# Anonymous Method
    * + - Anonymous method can be defined using the delegate keyword
        - Anonymous method must be assigned to a delegate.
        - Anonymous method can access outer variables or functions.
        - Anonymous method can be passed as a parameter.
        - Anonymous method can be used as event handlers
43. Nullable Type in C#
    * + - Nullable<T> type allows assignment of null to value types.
        - ? operator is a shorthand syntax for Nullable types.
        - Use value property to get the value of nullable type.
        - Use HasValue property to check whether value is assigned to nullable type or not.
        - Static Nullable class is a helper class to compare nullable types
44. var - Implicit typed local variable in C#
    * + - var can only be declared and initialized in a single statement
        - var cannot be used as a field type at the class level.
        - var cannot be used in an expression like var i += 10;
        - Multiple vars cannot be declared and initialized in a single statement
45. Func in C#
    * + - Func is built-in delegate type.
        - Func delegate type must return a value.
        - Func delegate type can have zero to 16 input parameters.
        - Func delegate type can be used with an [anonymous method](http://www.tutorialsteacher.com/csharp/csharp-anonymous-method) or [lambda expression](http://www.tutorialsteacher.com/linq/linq-lambda-expression).
46. Action Delegate
    * + - Action delegate is same as func delegate except that it does not return anything. Return type must be void.
        - Action delegate can have 1 to 16 input parameters.
        - Action delegate can be used with anonymous methods or lambda expressions
47. C# Anonymous Type
    * + - Anonymous type can be defined using the new keyword and object initializer syntax.
        - The implicitly typed variable- var, is used to hold an anonymous type.
        - Anonymous type is a reference type and all the properties are read-only.
        - The scope of an anonymous type is local to the method where it is defined.
48. C# Dynamic Type
    * + - The dynamic types are resolved at runtime instead of compile time.
        - The compiler skips the type checking for dynamic type
        - A method can have parameters of the dynamic type.
        - An exception is thrown at runtime if a method or property is not compatible
49. C# Object Initializer Syntax
    * + - A new way to initialize an object of a class or collection
        - Object initializers allow you to assign values to the fields or properties
        - Readable code.
        - Useful in multi-threading.

# 3 Tiers & 3 layers

1. What is tier
   * + - Tier indicates a physical separation of components
       - Layer indicates logical separation of components
2. Three Tier/Layer Architecture Design Components
   * + - We can separate the three tiers as Data Tier, Business Tier and Presentation Tier
       - Since each component is independent of each other
       - They are easily maintainable without changing the whole code
3. Data Tier
   * + - Data Tier is basically the server which stores all the application’s data
       - Data tier contents Database Tables, XML Files and other means of storing Application Data
4. Business Tier
   * + - Business Tier is mainly working as the bridge between Data Tier and Presentation Tier
       - All the Data passes through the Business Tier before passing to the presentation Tier
       - Business Tier is the sum of Business Logic Layer
       - Data Access Layer and Value Object and other components used to add business logic
5. Presentation Tier
   * + - Presentation Tier is the tier in which the users interact with an application
       - Presentation Tier contents Shared UI code
       - Code Behind and Designers used to represent information to user
6. Database Access Layer
   * + - Database Access Layer (DAO) builds the query based on received parameters
       - DAO passes it the dbConnection class for execution
       - DAO returns results from the dbConnection class to Business Logic Layer
7. Value Object
   * + - Value Object is nothing more but a class with the contents GET and SET methods
       - It’s mainly used to pass Data from one class to another
       - It’s directly connected with Business Logic Layer and Presentation Layer
8. Business Logic Layer
   * + - Business Logic Layer (BUS) works as a bridge between Presentation Layer and DAO
       - All the user values received from the presentation layer are being passed to BUS
       - The results received from the DAO are in row data in Data Table format but in BUS it’s converting into Value Objects (VO)
       - BUS is the most important class in the whole architecture
9. Presentation Layer
   * + - Presentation Layer is the only layer which is directly connected with the user
       - Presentation Layer is a really important layer for marketing purposes
       - Presentation Layer is mainly used for getting user data and then passing it to Business Logic Layer

# EF

1. What is Entity Framework
   * + - Entity Framework (EF) is an open source ORM framework for ADO.NET
       - An ORM takes care of creating database connections and executing commands
       - An ORM also helps to keep track of changes to those objects, and when instructed
2. Why Entity Framework
   * + - Entity Framework can generate the necessary database commands
       - You can express your queries against your domain objects using LINQ to entities
       - Entity Framework will execute the relevant query in the database and then materialize results into instances
       - Entity Framework is recommended data access technology for new applications.
       - ADO.NET seems to refer directly to the technology for data sets and data tables.
       - Microsoft recommends that you use Entity Framework over ADO.NET or LINQ to SQL for all new development
3. Conceptual Model
   * + - With Entity Framework, the focal point is referred to as a conceptual model
       - Your conceptual model may happen to align with your database schema
       - You can use a Visual Designer to define your conceptual model.
       - You can just define your classes and use a feature of Entity Framework
4. What is Features in Entity?
   * + - Entity Framework is a Microsoft tool.
       - Entity Framework is being developed as an Open Source product.
       - Entity Framework is no longer tied or dependent to the .NET release cycle.
       - Works with any relational database with valid Entity Framework provider.
       - SQL command generation from LINQ to Entities.
       - Entity Framework will create parameterized queries.
       - Tracks changes to in-memory objects.
       - Allows to insert, update and delete command generation.
       - Works with a visual model or with your own classes.
       - Entity Framework has stored Procedure Support.
5. Entity Framework – Architecture
   * + - Data Providers
       - Entity Client
       - Object Service
6. Entity Client
   * + - The Storage Layer contains the entire database schema in XML format.
       - The Entity Layer which is also an XML file defines the entities and relationships.
       - The Mapping layer is an XML file that maps the entities and relationships
       - The Metadata services which is also represented in Entity Client
       - The Metadata services provides centralized API to access metadata
7. Object Service
   * + - The main use of the Object Context is to perform different operations
       - It is the ORM layer of Entity Framework
       - This services allow developer to use some of the rich ORM features

# LINQ

1. What is LINQ?
   * LINQ (Language Integrated Query) is uniform query syntax.
   * LINQ is used to save and retrieve data from different sources
     + - LINQ always works with objects
       - You can use the same basic coding patterns to query and transform data
2. What are Types LINQ?
   * LINQ to Objects
   * LINQ to Dataset
   * LINQ to XML
   * LINQ to Entities
   * LINQ to SQL
   * By implementing IQueryable
3. Why LINQ?
   * Familiar language
   * Less coding
   * Readable code
   * Standardized way of querying multiple data sources
   * Compile time safety of queries
   * IntelliSense Support
   * Shaping data
4. What is LINQ API?
   * Use **System.Linq** namespace to use LINQ
   * LINQ api includes two main static class Enumerable & Queryable
5. Enumerable
   * The static **Enumerable** class includes extension methods for classes
   * Enumerable implements IEnumerable<T> interface
   * IEnumerable<T> type of collections
6. Queryable
   * The static **Queryable** class includes extension methods for classes
   * Queryable implements IQueryable<T> interface
   * Remote query provider implements IQueryable<T>
7. LINQ Query Syntax
   * **Query Syntax** is same like SQL (Structure Query Language) syntax
   * Query Syntax starts with from clause
   * Query Syntax can be end with Select or GroupBy clause
   * Use various other operators to construct the desired result
   * [Implicitly typed variable - var](http://www.tutorialsteacher.com/csharp/csharp-var-implicit-typed-local-variable) can be used to hold the result
8. What is keyword “Where” in LINQ?
   * **Where** is used for filtering the collection based on given criteria.
   * **Where** extension method has two overload methods.
   * Method Syntax requires the whole lambda expression
   * Multiple Where extension methods are valid in a single LINQ query
9. What is keyword “**OfType**” in LINQ?
   * The **Where** operator filters the collection based on a predicate function.
   * The **OfType** operator filters the collection based on a given type
   * **Where** and **OfType** extension methods can be called multiple times
10. “OrderBy” in LINQ:
    * LINQ includes five sorting operators
    * LINQ query syntax does not support OrderByDescending, ThenBy, ThenByDescending and Reverse.
    * LINQ only supports 'Order By' clause with 'ascending' and 'descending'
    * LINQ query syntax supports multiple sorting fields
11. ThenBy & ThenByDescending in LINQ
    * OrderBy and ThenBy sort collections in ascending order by default.
    * ThenBy or ThenByDescending is used for second level sorting in method syntax.
    * ThenByDescending method sorts the collection in descending order on another field.
    * ThenBy or ThenByDescending is NOT applicable in Query syntax.
    * Apply secondary sorting in query syntax by separating fields using comma.
12. GroupBy & ToLookup in LINQ
    * GroupBy & ToLookup return a collection.
    * The execution of GroupBy is deferred whereas that of ToLookup is immediate.
    * A LINQ query syntax can be end with the GroupBy or Select clause.
13. Join in LINQ
    * **Join** is like inner join of SQL
    * **Join** operates on two sequences inner sequence and outer sequence
    * **Join** query syntax
14. Quantifier in LINQ
    * All, Any & Contains are quantifier operators in LINQ.
    * All checks if all the elements in a sequence satisfies the specified condition.
    * Any check if any of the elements in a sequence satisfies the specified condition
    * Contains operator checks whether specified element exists in the collection or not.
    * Use Contains in IEqualityOperator to check for the object in the collection.
    * All, Any & Contains are not supported in query syntax in C# or VB.Net
15. Single & SingleOrDefault in LINQ
    * Single() expects one and only one element in the collection.
    * SingleOrDefault() will return default value of a data type of generic collection
    * Single() and SingleOrDefault() throw an exception when the collection not only one element
16. SequenceEqual in LINQ
    * The SequenceEqual method compares the number of items and their values for primitive data types.
    * The SequenceEqual method compares the reference of objects for complex data types.
    * Use IEqualityComparer class to compare two collection of complex type using SequenceEqual method.

# ASP.Net

1. What is ASP?
   * ASP stands for Active Server Pages
   * ASP is a Microsoft Technology
   * ASP is a program that runs inside a web server
2. What is ASP.NET?
   * ASP.NET is a web development platform
   * provides a programming model to build up web applications
   * ASP.NET works on top of the HTTP protocol
   * It uses the HTTP commands and policies to set a browser-to-server
   * ASP.NET is a part of Microsoft .Net platform
   * ASP.NET applications are compiled codes in .Net framework
   * These codes can use the entire hierarchy of classes in .Net framework
   * ASP.NET is used to produce interactive, data-driven web applications over the internet
   * It consists of a large number of controls
3. What is an ASP File?
   * An ASP file has the file extension ".asp"
   * An ASP file is just the same as an HTML file
   * An ASP file can contain server scripts in addition to HTML
   * Server scripts in an ASP file are executed on the server
4. What can ASP do for you?
   * Edit, change, add content, or customize any web page
   * Respond to user queries or data submitted from HTML forms
   * Access databases or other server data and return results to a browser
   * Provide web security since ASP code cannot be viewed in a browser
   * Offer simplicity and speed
5. How does ASP Work?
   * When a browser requests a normal HTML file, the server just returns the file.
   * When a browser requests an ASP file, the server passes the request to the ASP engine which reads the ASP file and executes the server scripts in the file.
   * Finally the ASP file is returned to the browser as plain HTML.
6. What is ASP.NET?
   * ASP.NET is a web development platform
   * provides a programming model to build up web applications
   * ASP.NET works on top of the HTTP protocol
   * It uses the HTTP commands and policies to set a browser-to-server
   * ASP.NET is a part of Microsoft .Net platform
   * ASP.NET applications are compiled codes in .Net framework
   * These codes can use the entire hierarchy of classes in .Net framework
   * ASP.NET is used to produce interactive, data-driven web applications over the internet
7. It consists of a large number of controls
   * User Input
   * Request.QueryString
   * Request.Form
   * Form Validation
8. What is a Cookie?
   * A cookie is often used to identify a user
   * A cookie is a small file that the server embeds on the user's computer
   * Each time the same computer requests a page with a browser, it will send the cookie too
   * With ASP, you can both create and retrieve cookie values
9. How to Create a Cookie?
   * The "Response.Cookies" command is used to create cookies
10. How to Retrieve a Cookie Value?
    * The "Request.Cookies" command is used to retrieve a cookie value
11. What if a Browser Does NOT Support Cookies?
    * Add parameters to a URL
    * Use a form
12. When does a Session Start?
    * A new user requests an ASP file, and the Global.asa file includes a Session\_OnStart procedure
    * A value is stored in a Session variable
    * A user requests an ASP file, and the Global.asa file uses the <object> tag to instantiate an object with session scope
13. When does a Session End?
    * A session ends if a user has not requested or refreshed a page
    * Use the Timeout property to set a timeout interval
14. Store and Retrieve Session Variables
    * You can store variables in Session object
    * Value in a session variable can be reached from ANY page
    * You can also store user preferences in the Session object
15. Remove Session Variables
    * The Contents collection contains all session variables.
    * It is possible to remove a session variable with the Remove method.
16. Loop through the Contents Collection
    * The Contents collection contains all session variables
17. Loop Through the StaticObjects Collection
    * You can loop Through the StaticObjects collectiont
18. ASP Application Object
    * Store and Retrieve Application Variables
    * Loop Through the Contents Collection
    * Loop Through the StaticObjects Collection
    * Lock and Unlock
19. What are content files that typically a project contains?
    * Page file (.aspx)
    * User control (.ascx)
    * Web service (.asmx)
    * Master page (.master)
    * Site map (.sitemap)
    * Website configuration file (.config)
20. Razor Syntax for C#
    * C# code blocks are enclosed in @{ ... }
    * Inline expressions (variables or functions) start with @
    * Code statements end with semicolon
    * Variables are declared with the var keyword
    * Strings are enclosed with quotation marks
    * C# code is case sensitive
    * C# files have the extension .cshtml
21. ASP.NET Web Pages - Page Layout
    * A Consistent Look
    * Content Blocks
    * Using a Layout Page
    * Preventing Files from Being Browsed
    * Hiding Sensitive Information
22. ASP.NET Web Pages - Folders
    * Logical Folder Structure
    * Physical Folder Structure
    * Virtual and Physical Names
    * URLs and Paths
    * The ~ Operator
    * The Server.MapPath Method
    * The Href Method
23. ASP.NET Web Pages - Global Pages
    * Before Web Startup: \_AppStart
    * Before Every Page: \_PageStart
24. ASP.NET Web Pages - Objects
    * The Page Object
    * Some Page Object Methods
    * Some Page Object Properties
    * The Page Property
25. ASP.NET Web Pages - Files
    * Working with Text Files
    * Add a Text File Manually
    * Displaying Data from a Text File
    * Displaying Data from an Excel File
26. ASP.NET Razor - C# Variables
    * Variables
    * Data Types
    * Operators
    * Converting Data Types
27. ASP.NET Razor - C# Loops and Arrays
    * For Loops
    * For Each Loops
    * While Loops
    * Arrays
28. ASP.NET Razor - C# Logic Conditions
    * The If Condition
    * The Else Condition
    * The Else If Condition
    * Switch Conditions
29. ASP.NET Web Pages - Databases
    * Displaying Data from Database
    * Adding a Customers Page
    * ASP.NET Database Object Reference

# MVC

1. What is MVC?
   * MVC: Model, View and Controller
   * Controller: handles user's requests & renders appropriate View
   * View : user interface; displays the data
   * Model: data model or business model
2. Routing in MVC
   * Routing plays important role in MVC framework.
   * Routing maps URL to physical file or class (controller class in MVC).
   * Route contains URL pattern and handler information.
   * URL pattern starts after domain name.
   * Routes can be configured in RouteConfig class.
   * Multiple custom routes can also be configured.
   * Route constraints applies restrictions on the value of parameters.
   * Route must be registered in Application\_Start event in Global.ascx.cs file.
3. What is Controller?
   * A Controller handles incoming URL requests.
   * MVC routing sends request to appropriate controller
   * All the public methods in the Controller class are called Action methods.
   * A Controller class must be derived from System.Web.Mvc.Controllerclass
   * A Controller class name must end with "Controller".
   * New controller can be created using different scaffolding templates
   * You can create custom scaffolding template also.
4. What is Action method?
   * All the public methods in the Controller class are called Action methods.
   * Action method has following restrictions.
   * Action method must be public
   * Action method cannot be overloaded.
   * Action method cannot be a static method.
   * ActionResult is a base class of all the result type
   * Base Controller class contains methods that returns appropriate result type
   * Action method can include [Nullable](http://www.tutorialsteacher.com/csharp/csharp-nullable-types) type parameters
5. What are Action selectors?
   * Action Selectors attributes are used to determine which action method to invoke.
   * Three action selectors attributes are available in MVC 5
   * ActionName
   * NonAction
   * ActionVerbs
   * ActionName attribute is used to specify different name of action
   * NonAction attribute marks the public method of controller class as non-action method. It cannot be invoked.
6. What are ActionVerbs
   * ActionVerbs are another Action Selectors which selects an action method based on request methods
   * Multiple action methods can have same name with different action verbs
   * Multiple action verbs can be applied to a single action method
7. Model in ASP.NET MVC
   * Model represents domain specific data and business logic in MVC architecture
   * It maintains the data of the application
   * Model objects retrieve and store model state
   * Model class holds data in public properties
   * All the Model classes reside in the Model folder in MVC folder structure
8. View in ASP.NET MVC
   * View is a User Interface which displays data and handles user interaction.
   * Views folder contains separate folder for each controller.
   * ASP.NET MVC supports Razor view engine in addition
   * Razor view files has .cshtml or .vbhtml extension
9. Razor syntax
   * Use @ to write server side code.
   * Server side code block starts with @{\* code \* }
   * Use @: or <text></<text> to display text from code block.
   * if condition starts with @if{ }
   * for loop starts with @for
   * @model allows you to use model object anywhere in the view
10. HTTP Helpers
    * HtmlHelper extension method generates html elements based on model properties.
    * It is advisable to use "For" extension methods for compile time type checking
11. What are differences between TextBox and TextBoxFor?
    * @Html.TextBox() is loosely typed method
    * @Html.TextBoxFor() is a strongly typed (generic) extension method
    * TextBox() requires property name as string parameter
    * TextBoxFor() requires lambda expression as a parameter.
    * TextBox doesn't give you compile time error
    * TextBoxFor is generic method so it will give you compile time error
12. Model Binding
    * Previously bound action parameters, when the action is a child action
    * Form fields (Request.Form)
    * The property values in the JSON Request body (Request.InputStream)
    * Route data (RouteData.Values)
    * Querystring parameters (Request.QueryString)
    * Posted files (Request.Files)
13. How to implement Data Validation in MVC?
    * ASP.NET MVC uses DataAnnotations attributes for validation.
    * DataAnnotations attributes can be applied to the properties
    * The following validation attributes available by default
      + - * Required
          * StringLength
          * Range
          * RegularExpression
          * CreditCard
          * CustomValidation
          * EmailAddress
          * FileExtension
          * MaxLength
          * MinLength
          * Phone
    * Use ValidationSummary to display all the error messages in the view.
    * Use ValidationMessageFor or ValidationMessage helper method to display field level error messages in the view.
    * Check whether the model is valid before updating in the action method using ModelState.IsValid.
    * Enable client side validation to display error messages without postback effect in the browser
14. Layout View
    * The Layout view contains common parts of a UI
    * \_ViewStart.cshtml file can be used to specify path of layout page,
    * You can set the Layout property in the individual view also
    * Layout view uses two rendering methods: RenderBody() and RenderSection().
    * RenderBody can be used only once in the layout view
    * RenderSection method can be called multiple time with different name.
    * RenderBody method renders all the content of view
    * RenderSection method renders the content of a view
    * RenderSection can be configured as required or optional
15. Partial View
    * Partial view is a reusable view
    * Partial view can be rendered using Html.Partial(), Html.RenderPartial() or Html.RenderAction() method
16. ViewBag in ASP.NET MVC
    * ViewBag transfers data from the controller to the view
    * ViewBag is a dynamic property
    * You can assign any number of properties and values to ViewBag
    * The ViewBag's life only lasts during the current http request
    * ViewBag is actually a wrapper around ViewData
17. View Data
    * ViewData transfers data from the Controller to View
    * ViewData is derived from ViewDataDictionary
    * ViewData's life only lasts during current http request.
    * ViewData values will be cleared if redirection occurs.
    * ViewData value must be type cast before use.
    * ViewBag internally inserts data into ViewData dictionary
18. What is TempData?
    * TempData can be used to store data between two consecutive requests
    * TemData is a TempDataDictionary type.
    * TempData internaly use Session to store the data
    * TempData value must be type cast before use
    * TempData can be used to store only one time messages
    * Call TempData.Keep() to keep all the values of TempData in a third request
19. Filters in MVC
    * MVC Filters are used to execute custom logic before or after executing action method.
    * MVC Filter types:
      + - * Authorization filters
          * Action filters
          * Result filters
          * Exception filters
    * Filters can be applied globally in FilterConfig class
    * Custom filter class can be created by implementing FilterAttribute class and corresponding interface
20. Action Filters
    * Action filters allow pre and post processing logic
    * Action filters are generally used to apply cross-cutting concerns
    * Action filter can be registered as other filters
    * Custom action filter attribute can be created by deriving ActionFilterAttribute class or implementing IActionFilter interface and FilterAttribute abstract class.
    * Every action filter must override OnActionExecuted, OnActionExecuting, OnResultExecuted, OnResultExecuting methods.
21. Bundling in MVC
    * Bundling and Minification minimize static script or css files loading time
    * MVC framework provides ScriptBundle, StyleBundle and DynamicFolderBundle classes.
    * ScriptBundle does minification of JavaScript files.
    * StyleBundle does minification of CSS files.
22. ScriptBundle in ASP.NET MVC
    * Bundling and Minification minimize static script or css files loading time
    * ScriptBundle does minification of JavaScript files.
    * Create script or css bundles in BundleConfig class included in App\_Start folder.
    * Use wildcard {version} to render available version files at runtime.
    * Use Scripts.Render("bundle name") method to include script bundle in a razor view.