**C SHARP**

**C# - C Sharp**

It is an object-oriented programming language created by Microsoft that runs on the .NET Framework.

**C# is used for:**

* Mobile applications
* Desktop applications
* Web applications
* Web services
* Web sites
* Games
* Database applications

**Why we Use?**

* It is one of the most popular programming languages in the world
* It is easy to learn and simple to use
* It has a huge community support
* C# is an object-oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs.

**OOP’S - Object-Oriented Programming**

OOPs is writing procedures or methods that perform operations on the data and creating objects that contain both data and methods.

* It is faster and easier to execute
* It provides a clear structure for the programs
* It makes the code easier to maintain, modify and debug
* It is possible to create full reusable applications with less code and shorter development time

**Class**

A class is a template for objects.

**Object**

An object is an instance of a class.

**Inheritance**

It inherits fields and methods from one class to another.

* **Derived Class** (child) - the class that inherits from another class
* **Base Class** (parent) - the class being inherited from

**Polymorphism**

It provides the ability to a class to have multiple implementations with the same name.

**Abstraction**

It is the process of hiding certain details and showing only essential information to the user. To achieve security.

Abstraction can be achieved with either **abstract classes** or [**interfaces**](https://www.w3schools.com/cs/cs_interface.asp).

* **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
* **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the derived class (inherited from).

**Encapsulation**

Encapsulation is defined as the wrapping up of data under a single unit.

**Type Casting**

Type casting is when you assign a value of one data type to another type.

* **Implicit Casting** (automatically) - converting a smaller type to a larger type size.  
  char -> int -> long -> float -> double
* **Explicit Casting** (manually) - converting a larger type to a smaller size type.  
  double -> float -> long -> int -> char

**Method**

A **method** is a block of code which only runs when it is called.

**Method Overloading**

Creating more than one method or function having same name but different signatures or the parameters in the same class is called method overloading.

**Method Overriding**

Creating a method in the derived class with the same signature as a method in the base class is called as method overriding.

**Constructors**

Constructor is a special method which is invoked automatically at the time of object creation.

* **Default constructor -** A constructor which has no argument
* **Parameterized constructor -** A constructor which has parameters

**Destructors**

Destructors are usually used to deallocate memory and do other clean-up for a class object and its class members when the object is destroyed.

**Access Modifiers**

It is used to set the access level/visibility for classes, fields, methods and properties.

* **Public** - The code is accessible for all classes
* **Private -** The code is only accessible within the same class
* **Protected -** The code is accessible within the same class, or in a class that is inherited from that class
* **Internal -** The code is only accessible within its own assembly

**Enum**

It represents a group of **constants.**

**Arrays**

Arrays are used to store multiple values in a single variable.

**Jagged Arrays**

A jagged array is an array of array. Jagged arrays store arrays instead of literal values.

**Delegates**

A delegate is a reference type variable that holds the reference to a method. The reference can be changed at runtime. It is used for implementing events and the call-back methods.

**Events**

An event is a notification sent by an object to signal the occurrence of an action.

**Collections**

 Collection types are designed to store, manage and manipulate similar data more efficiently.

**Generics**

Generics defines the specification of the data type of programming elements in a class or a method.

|  |  |
| --- | --- |
| **Non-Generic** | **Generic** |
| ArrayList | List |
| HashTable | Dictionary |
| SortedList | SortedList |
| Stack | Stack |
| Queue | Queue |

**Boxing**

When a value type is converted to object type, it is called boxing.

**Un Boxing**

When a value type is converted to value type, it is called un- boxing.

**Sealed Class**

When a class is declared sealed, it cannot be inherited.

**ENTITY FRAMEWORK**

**Entity Framework**

Entity Framework is an **Object Relational Mapper (ORM)** which is a type of tool that simplifies mapping between objects in your software to the tables and columns of a relational database.

Entity Framework is the best way to develop database applications. I used to develop my applications using LINQ to SQL.

**Is entity framework better than ado net?**

**ADO.NET** provides better performance as it is directly connected to the data source, which makes the processing faster than **Entity Framework** as it translates LINQ queries to SQL first then process the query.

**Differentiate ADO.NET from Entity Framework.**

|  |  |
| --- | --- |
| **ADO.NET** | **ENTITY FRAMEWORK** |
| ADO.NET is fast. | Entity Framework is comparatively slower. |
| It creates several data layer codes. | It does not create any data layer codes. |
| It does not create codes for the data access layers, intermediate layers, and mapping codes by itself. | It automatically creates codes for the data access layers, intermediate layers, and mapping codes. This eventually helps developers cut down the development of work and time. |

**Primary functions of EF**

* It helps map domain classes to the database schema translates.
* It keeps tracks of changes in the entities.
* It helps execute LINQ queries to SQL.
* It stores the changes stats to the database.

**Advantages**

* It provides a wide range of prototypes that helps write object-oriented codes.
* It includes auto migration support that helps configure or manage a database easily and quickly.
* It provides many alternate commands that help shorten codes and make a coding job a lot easier.

**Disadvantages**

* It is considered a slower form of ORM.
* If a user does not use raw SQL codes, things might become difficult.
* It directs the shape of the entire model when it comes to difficult projects. Additionally, cleaning a model is not possible without an ORM technique.

**Different types of approaches used in Entity Framework**

* Model First Approach
* Code First Approach
* Database First Approach

**Differentiate LINQ from Entity Framework**

|  |  |
| --- | --- |
| **LINQ** | **ENTITY FRAMEWORK** |
| It only operates with the help of the SQL Server Database. | It has various databases, such as SQL Server, MYSQL, Oracle, DB2, etc. |
| It maintains a relation by creating a .dbml file. | It first creates the .edmx file. |
| It cannot generate a database by using the model. | It can generate a database using the model. |
| It supports one to one mapping between entity classes and the relational databases | It supports one to one, one to many, and many to many mapping between entity classes and the relational databases |
| It enables users to query the data with DataContext | It enables users to query the data with DbContext |

**ASP.NET MVC**

**ASP.NET**

ASP.NET MVC is an open-source software from Microsoft. Its web development framework combines the features of MVC (Model-View-Controller) architecture.

## **MVC (Model View Controller)**

Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces.

MVC is a framework for building web applications using an MVC (Model View Controller) design:

* The Model is the part of the application that handles the logic for the application data.
* The View displays the data (the database records).
* The Controller handles the user interaction (to the database records).

**The MVC model defines web applications with 3 logic layers,**

* The business layer (Model logic)
* The display layer (View logic)
* The input control (Controller logic)

**Advantages of MVC**

* Faster development process: MVC supports rapid and parallel development.
* Ability to provide multiple views.
* Support for asynchronous technique.
* The modification does not affect the entire model.
* MVC model returns the data without formatting.

**Filters**

Filter is a custom class where you can write custom logic to execute before or after an action method executes

* **Authorization Filters**

Performs authentication and authorizes before executing an action method.

* **Action filters**

Performs some operation before and after an action method executes.

* **Result filters**

Performs some operation before or after the execution of the view.

* **Exception filters**

Performs some operation if there is an unhandled exception thrown during the execution of the ASP.NET MVC pipeline.

**ViewData**

ViewData is used to pass data from controller to view.

**ViewBag**

ViewBag is also used to pass data from the controller to the respective view.

**TempData**

TempData is used to pass data from the current request to the next request

**Partial View**

A partial view is a reusable portion of a web page. It can be used in one or more [Views](https://www.tutorialsteacher.com/mvc/mvc-view) or [Layout Views](https://www.tutorialsteacher.com/mvc/layout-view-in-asp.net-mvc).

**Routing**

Routing is the process of directing an HTTP request to a controller and the functionality of this processing.

**ANGULAR**

**ANGULAR**

Angular is a TypeScript-based [open-source web application framework](https://hackr.io/blog/top-10-web-development-frameworks-in-2020), developed and maintained by Google. It offers an easy and powerful way of building front end web-based applications

**ADVANTAGES**

* Ability to add a custom directive
* Exceptional community support
* Facilitates client and server communication
* Features strong features, such as Animation and Event Handlers
* Follows the MVC pattern architecture
* Offers support for static template and Angular template
* Support for two-way databinding
* Supports dependency injection, RESTful services, and validations

**DISADVANTAGES**

* Complex SPAs can be inconvenient and laggy to use due to their size
* Dynamic applications do not always perform well
* Learning Angular requires a decent effort and time

|  |  |
| --- | --- |
|  |  |