|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| <http://csharptopicwiseques.blogspot.in/2010/08/wcf.html> 1. What is Object Oriented Programming? OOP is a technique to develop logical modules, such as classes that contains properties, fields and events. OOP provides many concepts such as inheritance, data binding, polymorphism etc.  Object means a real word entity such as pen, paper, chair, table etc.  Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects.  It simplifies the software development and maintenance by providing some concepts:   * Object * Class * Inheritance * Polymorphism * Abstraction * Encapsulation  2. What is a class? A class can be defined as the primary building block of OOP. A class contains data and behavior of an entity.  A class in C# can contain:  1. Data member 2. Properties 3. Constructor 4. Methods  Notes:   1. Class name should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc. 2. The name of the constructor is always same as the class name 3. A class can have any number of data members, properties, constructors and methods 4. Data member defined using a class is called as object reference. 5. A class can have a data member which is an object reference of the same class Like the manager of the employee is also an employee.  3. What is an Object? An entity that has state and behavior is known as an object e.g. pen, table, car etc. It can be physical or logical.  An object has three characteristics:   * State: represents data (value) of an object. * Behavior: represents the behavior (functionality) of an object such as deposit, withdraw etc. * Identity: Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the compiler to identify each object uniquely.   For Example: Pen is an object. Its name is Parker, color is black etc. known as its state. It is used to write, so writing is its behavior. Object is an instance of a class. Class is a template or blueprint from which objects are created. So object is the instance (result) of a class. For example, you have a class called Vehicle and car is the object of that class. 4. Explain the basic features of OOPs? OOPs have four basic features.   * Abstraction : Abstraction is the process of exposing only the relevant data to the users without showing unnecessary information * Polymorphism : It allows you to use an entity in multiple forms * Encapsulation : Prevents the data from unwanted access by binding of code and data in a single unit called object * Inheritance: Promotes the reusability of code and eliminates the use of redundant code.  5. What are Abstract classes? An abstract class is a class that cannot be instantiated and is always used as a base class.  Characteristics of an abstract class:   * You cannot instantiate an abstract class directly * You can have abstract as well as non-abstract members in an abstract class * You must declare one abstract method in the abstract class * An abstract class is always public * An abstract class is declared using abstract keyword  6. Explain the features of an Interface  * An Interface contains only the signature of methods * An Interface has no Implementation on its own * An Interface is used to implement multiple inheritance in code. * It defines a static set of methods and their arguments * Variables in Interface must be declared as public, static and final * Methods in an Interface must be declared as public and abstract * A class implementing an Interface must implement all of its method * An Interface can derive from more than one Interface  7. Difference between an abstract class and an Interface?  |  |  | | --- | --- | | Abstract Class | Interface | | A class can extend only one Abstract class | A class can  implement several  interfaces | | The member of an abstract class can be private  and protected | An Interface can  only have public  members | | Abstract classes should have subclasses | Interfaces must  have Implementations  by classes | | Any class can extend an abstract class | Only an Interface  can extend another  Interface | | Methods in an abstract class can be abstract as well as concrete | All methods  in an Interface  should be abstract | | There can be a constructor for Abstract class | Interface does not have constructor | | An abstract class can Implement methods | Interfaces cannot contain body  of any of its method |  8. Explain different types of Inheritance There are four types of Inheritance in Object oriented programming.   * Single Inheritance : contains one base class and one derived class * Hierarchical Inheritance : Contains one base class and multiple derived classes of same base class * Multilevel Inheritance : Contains a class derived from a derived class * Multiple Inheritance : Contains several base class and a derived class  9. What is Constructor? C# constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object that is why it is known as constructor. A constructor is a special method that is used to initialize an object. Every class has a constructor, if we don’t explicitly declare a constructor for any C# class the compiler builds a default constructor for that class. A constructor does not have any return type. Constructors are responsible for object initialization and memory allocation of its class.  There are basically two rules defined for the constructor.  1. Constructor name must be same as its class name 2. Constructor must have no explicit return type  You can read more on constructor [here](http://www.csharpstar.com/csharp-constructor/). 10. What is Destructor? A Destructor is automatically invoked when an object is finally destroyed. The name of the Destructor is same as class and prefixed with a tilde (~)  A Destructor is used to free the dynamic allocated memory and release the resources. You can Implement a custom method that allows controlling object destruction by calling the destructor.   * Destructors don not have any return type * Destructors are always public * Destructors cannot be overloaded  11. What is a Static Constructor? Static constructor is used to initialize static data of a class. Compiler calls the static constructor before the first instance is created.   * No access specifier is required to define it * You cannot pass parameters to static constructor * A class can have only one static constructor * It can access only static members of class * It is invoked only once, when the program execution begins  12. What is Method Overloading? Method Overloading lets you have 2 methods with same name and different signature. Overloading can be achieved: -By changing the number of parameters used. -By changing the order of parameters. -By using different data types for the parameters. 13. What is Access Specifier? An access specifier defines the scope and visibility of a class member. C# supports the following access specifiers:   * Public * Private * Protected * Internal * Protected internal  14. What is Encapsulation?  * Encapsulation is a process of hiding the members from outside of a class and implemented using access specifiers * Encapsulation is also called as information hiding. * Encapsulation provides a way to preserve the integrity of state data. Rather than defining public fields, private data fields should be defined. * Well-encapsulated class should hide its data and the details of how it operates on data from the outside world. This is termed black box programming. * Using this, implementation of the method can be changed by the class author without breaking any existing code making use of it.  15. What is the difference between Method Overloading and Method Overriding?  |  |  | | --- | --- | | **Method Overloading** | **Method Overriding** | | Method Overloading lets you have 2 methods with same name and different signature | Method Overriding lets  you have 2 methods  with same name and  same signature | | Overloading is called as compile time polymorphism or early binding | Overriding is called as run time  polymorphism or late binding or dynamic  polymorphism | | Overloading can be achieved: -By changing the number of parameters used. -By changing the order of parameters. -By using different data types for the parameters. | Overriding can be achieved: -Creating the method in  a derived class with same  name, same parameters and  same return type as in base  class is called as method  overriding | | Method overloading can be overloaded in same class or in the child class. | Method overriding is only possible in  derived class not within the same class  where the method is declared | | **Example:**  Public class test  {  public virtual int addition()  {  return 10;  }  }  public class Amount:test  {  public override int balance()  {  return 500;  }  } |  |  |  |  | | --- | --- | |  |  |  16. What is the difference between Abstraction and Encapsulation? Encapsulation is wrapping, just hiding properties and methods. Encapsulation is used for hide the code and data in a single unit to protect the data from the outside the world. Class is the best example of encapsulation. Abstraction refers to showing only the necessary details to the intended user. 17. What is Operator Overloading? Operator overloading is a technique by which operators used in a programming language are implemented in user-defined types with customized logic that is based on the types of arguments passed.  Operator overloading facilitates the specification of user-defined implementation for operations wherein one or both operands are of user-defined class or structure type. This helps user-defined types to behave much like the fundamental primitive data types. Operator overloading is helpful in cases where the operators used for certain types provide semantics related to the domain context and syntactic support as found in the programming language. It is used for syntactical convenience, readability and maintainability 18. What is a Delegate? Delegates are type safe function pointer. It holds reference to a function.  The signature of delegate matches the signature of the function that it points to else you will get compilation error. Delegates are type safe pointers because it points to a function and holds the signature of the function. 19. What is multicast Delegate? A multicast delegate is a delegate that has references to more than one function. When you invoke a multicast delegate, all the functions the delegate is pointing to are also invoked.  There are 2 ways to create multicast delegate.  + or += to register a method with the delegate – or -= to unregister a method with the delegate A multicast delegate invokes the methods in invocation list in same order in which they are added. 20. What are Events? Events are user actions such as key press, clicks, mouse moves, etc., or some occurrence such as system generated notifications. Applications need to respond to events when they occur. 21. What is the difference between Array and Collection?  |  |  | | --- | --- | | Array | Collection | | You need to specify the size of an array at the time of its declaration. It cannot be resized. | The size of a collection can be adjusted  Dynamically as per user’s requirement.  It does not have fixed size | | The member of an array should be of the same data type | Collection can have elements of different  types |  22. What is the difference between Shadowing and Overriding? Shadowing:  1. You can shadow a base class member in the derived class by using the keyword New. 2. The method signature, access level and return type of the shadowed member can be completely different than the base class member. 3. A method or function of the base class is available to the child (derived) class without the use of the “overriding” keyword. 4. The compiler hides the function or method of the base class. This concept is known as shadowing or method hiding. 5. In the shadowing or method hiding, the child (derived) class has its own function, the same function is also available in the base class.  Overriding:  1. Method overriding is an important feature of OOPS that allows us to re-write a base class function or method with a different definition. 2. Overriding is also known as “Dynamic Polymorphism” because overriding is resolved at runtime. 3. The method signature, access level and return type of the hidden member has to be same as the base class member 4.In other words both methods (base class method and derived class method) have the same name, same number and same type of parameter in the same order with the same return type. 5. The overridden base method must be virtual, abstract or override.  You can look at C# implementation on Shadowing and Overriding [here](http://www.csharpstar.com/overriding-vs-shadowing-in-csharp/). 23. What is the difference between Class and Structure?  |  |  | | --- | --- | | Class | Structure | | A Class is a reference type | A Structure is a value type | | By default, the members of a Class are private | By default, the members of a Structure are public | | Class supports Inheritance | Structure does not support Inheritance | | Class can contain constructor/destructor | Structure does not require Constructor/Destructor | | Variables of a Class can be assigned as null | Structure members cannot have null values |  24. What is the similarities between Class and Structure?  * Access specifiers are identically used in structure and classes to restrict the access of their data and methods outside their body * Both can have constructors, methods, properties, fields, constants etc. * Both can implement Interfaces to use multiple-inheritance in code * Both can have Delegates and Events  25. What is Enum?  * An Enum is a distinct value type with a set of named constants * It is declared by using keyword Enum * Every Enum type has an underlying type which can be any integral type except char. The default underlying type is int. * The first enumerator has the value 0 by default and the value of each successive enumerator is increased by 1 * Enumeration bridge the gap between numbers and objects  26. What is a Nested Class? Nested class is nothing but defining a class within another class.  Nested classes has the ability to specify private as an access modifier for the class itself.The use of the private access modifier defines the intended accessibility of the class and prevents access from outside the class. 27. What is an Indexer? An Indexer is a member that enables an object to be indexed like Arrays. 28. What is Sealed keyword in C#? You can disable inheritance by using the sealed keyword on a class or a method. When used on a class, you can’t derive other classes from it. When used on a method, derived classes can’t override the method. 29. What is a hashtable? Hashtable is store multiple items and each of these items is associated with unique string key. Each item can be accessed using the key associated with it. 30. What is Polymorphism and explain different types of Polymorphism? The word polymorphism means having many forms. Polymorphism is ‘one interface, multiple functions’. Polymorphism can be static or dynamic.  Static Polymorphism:  The mechanism of linking a function with an object during compile time is called early binding. It is also called static binding. C# provides two techniques to implement static polymorphism. They are:  1.Function overloading 2.Operator overloading  Dynamic Polymorphism:  The mechanism of linking method to an object during run time is called late binding.It is also known as run-time polymorphism. Method overriding helps to implement Dynamic Polymorphism. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1. What is ASP.net MVC or Explain the overview of MVC (Model-View-Controller)?**  MVC (Model-View-Controller) was creates to support a pattern based software development.By using MVC pattern,  We can develop applications that are more flexible to changes without affecting the other components of our application.   * “**Model**”, is for data. It contains all application validation/business logic that is not contained in view or controller. * “**View**”, contains HTML markup and view logic * “**Controller**”, contains control flow logic. It interacts with mvc model and views to control the flow of application execution.   [ASP.net MVC](http://www.csharpstar.com/wp-content/uploads/2015/11/mvc1.jpg)   2. Difference between ASP.NET MVC and ASP.NET WebForms? ASP.NET Web Forms uses Page controller pattern approach for rendering layout. In this approach,  every page has its own controller i.e. code-behind file that processes the request. On the other hand,  ASP.NET MVC uses Front Controller approach. In this approach a common controller for all pages,  Processes the requests.   |  |  | | --- | --- | | **ASP.net web form** | **ASP.net MVC** | | Page controller pattern that means an implicit controller (code behind) would process the request | Front controller pattern that means  An explicit controller would  be there to  process the request | | web form has user controls for code reusability | Partial views has code reusability | | view and controllers are not separated | view and controller are handled separately | | stateful and view state is used to maintain state | Stateless, so view state is not used | | Has server control | Has html helper | | views are tightly coupled with business logic | views and logics are separately managed | | Master pages for constant look and feel | MVC layouts for constant look and feel | | Filebased urls and it needs physical files | Route based urls and doesnot need  a physical file. It depends on controller | | Recommended for small scale applications | Recommended for large scale applications |        3. What are the Core features of ASP.NET MVC? Core features of ASP.net MVC framework are:   * **Clear separation of application concerns** (Presentation and Business Logic).   It reduces complexity that makes it ideal for large scale applications where multiple teams are working.   * It’s an **extensible** as well as **pluggable framework***.* We can plug components and further customize them   easily.   * It provides extensive support for URL Routing that helps to make friendly URLs (means friendly for human   as well as Search Engines).   * It supports for **Test Driven Development (TDD)** approach. In ASP.NET WebForms, testing support is   Dependent on Web Server but ASP.NET MVC makes it independent of Web Server, database or any other classes.   * Support for **existing ASP.NET features** like membership and roles, authentication and authorization,   Provider model and caching etc.   4. What all are the advantages of ASP.Net MVC? The ASP.NET MVC framework offers the following advantages:   * It makes it easier to manage complexity by dividing an application into the model, the view, and the controller. * It does not use view state or server-based forms. It gives full control over the behavior of an application. * It uses a Front Controller pattern that processes Web application requests through a single controller. * It provides better support for test-driven development (TDD). * It works well for Web applications that are supported by large teams of developers and Web designers   Who need a high degree of control over the application behavior?     5. What is the difference between ViewData, ViewBag and TempData? [view bag & View Data](http://www.csharpstar.com/wp-content/uploads/2015/11/mvc7.jpg)     * Both **ViewBag and ViewData** are used to to communicate between controller and corresponding view. * It’s a mechanism to maintain state between controller and corresponding view. * **ViewData** is a dictionary object while **ViewBag** is a dynamic property (a new C# 4.0 feature). * You can use View Data to represent any type of information including strings,objects and database records * ViewBag doesn’t have typecasting and null checks. * **TempData** is a dictionary object that stays for the time of an HTTP Request. So, Tempdata   Can be used to maintain data between redirects i.e from one controller to the other controller.  For more detailed understanding, read this [**article**](http://www.csharpstar.com/difference-between-viewdata-viewbag-and-tempdata/).      **6. What is Routing in ASP.NET MVC?**    [ASP.net MVC Routing](http://www.csharpstar.com/wp-content/uploads/2016/02/MVC_Routing.jpg)  Routing plays an important role in an ASP.NET MVC Application execution flow.  It maps request URL to a specific controller action using a Routing Table.  In order to describe user’s actions, MVC framework uses friendly URLs against actions instead of mapping it to Physical files as in case of an asp.net Web Form application.  You can read the detailed article on Routing [**here**](http://www.csharpstar.com/routing-in-asp-net-mvc/).      **7. What are Action Methods in ASP.NET MVC?**    [ASP.net MVC Controller](http://www.csharpstar.com/wp-content/uploads/2016/02/MVC_Controller.jpg)  Action methods perform certain operations by interacting with Model and return output data back to View.  Return type of an action method in ASP.NET MVC is called an action result.  ASP.NET MVC framework provides an abstract class named “ActionResult” which is basically a base class for all action results. Also, there are many built-in action result types derived from ActionResultclass like ViewResult,  PartialViewResult, RedirectResult etc. You can read more on controller and Action methods [**here**](http://www.csharpstar.com/controllers-and-action-methods-in-asp-net-mvc/).    **8. What are Action Filters in ASP.NET MVC?**   * Action filters are used when you want to apply some specific logic before or after action methods. * You can apply these action filters to a controller or a specific controller action. * Action filters are basically custom classes that provide a mean for adding pre-action or post-action behavior to   Controller actions. Types of Action Filters: [ACtion Filters](http://www.csharpstar.com/wp-content/uploads/2016/02/MVC_Action_Filter.jpg)    **Authorization** filters are used to implement authentication and authorization for controller actions.  **Action** filters contain logic that is executed before and after a controller action executes. You can use an action filter, for instance, to modify the view data that a controller action returns.  **Result** filters contain logic that is executed before and after a view result is executed. For example, you might want to modify a view result right before the view is rendered to the browser.  **Exception** filters are the last type of filter to run. You can use an exception filter to handle errors raised by either your controller actions or controller action results. You also can use exception filters to log errors.  The base class for all action filters is the System.Web.Mvc.FilterAttribute.   1. Authorization filters – Implements the <span style="font-family: Courier New;">IAuthorizationFilter</span> attribute. 2. Action filters – Implements the <span style="font-family: Courier New;">IActionFilter</span> attribute. 3. Result filters – Implements the <span style="font-family: Courier New;">IResultFilter</span> attribute. 4. Exception filters – Implements the <span style="font-family: Courier New;">IExceptionFilter</span>attribute.  Base ActionFilterAttribute Class:  * To implement a custom action filter, the ASP.NET MVC framework includes a base <span style="font-family: Courier New;">ActionFilterAttribute</span> class. * This class implements both the <span style="font-family: Courier New;">IActionFilter</span> and <span style="font-family: Courier New;">IResultFilter</span> interfaces and inherits from the <span style="font-family: Courier New;">Filter</span> class. * The base <span style="font-family: Courier New;">ActionFilterAttribute</span> class has the following methods that you can override:   + OnActionExecuting – This method is called before a controller action is executed.   + OnActionExecuted – This method is called after a controller action is executed.   + OnResultExecuting – This method is called before a controller action result is executed.   + OnResultExecuted – This method is called after a controller action result is executed.   **Example**:  Let’s take an example to create a LogActionFilter to understand it better.   |  |  | | --- | --- | |  | using System;  using System.Diagnostics;  using System.Web.Mvc;  using System.Web.Routing;  namespaceMvcApplication1.ActionFilters  {  publicclassLogActionFilter:ActionFilterAttribute       {  public override void OnActionExecuting(ActionExecutingContext filterContext)            {                 Log("OnActionExecuting",filterContext.RouteData);            }            Public override void OnActionExecuted(ActionExecutedContext filterContext)            {                 Log("OnActionExecuted",filterContext.RouteData);            }            publicoverride voidOnResultExecuting(ResultExecutingContext filterContext)            {                 Log("OnResultExecuting",filterContext.RouteData);            }            publicoverride voidOnResultExecuted(ResultExecutedContext filterContext)            {                 Log("OnResultExecuted",filterContext.RouteData);            }            privatevoidLog(stringmethodName,RouteData routeData)            {                 varcontrollerName=routeData.Values["controller"];                 varactionName=routeData.Values["action"];                 varmessage=String.Format("{0} controller:{1} action:{2}",methodName,controllerName,actionName);                 Debug.WriteLine(message,"Action Filter Log");            }       }  } | |  |  |   In the above example, the OnActionExecuting(), OnActionExecuted(), OnResultExecuting(), and OnResultExecuted() methods all call the Log() method. The name of the method and the current route data is passed to the Log() method. The Log() method writes a message to the Visual Studio Output window. [Output on Action Filter in ASP.net MVC](http://techkatak.com/wp-content/uploads/2015/08/RunActionFilter.jpg)  Let’s see how to apply action filter:   |  |  | | --- | --- | |  | using System.Web.Mvc;  using MvcApplication1.ActionFilters;  namespaceMvcApplication1.Controllers  {  [LogActionFilter]       publicclassHomeController:Controller       {  publicActionResult Index()            {                 returnView();            }            publicActionResult About()            {                 returnView();            }       }  } | |  |  |     **9. Can you explain briefly the Authorize attribute and its significance in ASP.net MVC?**  Traditionally in ASP.NET we achieve these concept by isolating critical modules from the rest of the application, i.e. by segregating ASPX pages in a folder under the control of a custom web.config file that redirects not-authenticated and unauthorized users to a custom login page.  Now **ASP.NET MVC** relieves the pain in attaining the role based security just by a simple powerful attribute known as **Authorize.**  In default all the Controllers and Action methods are accessible by both Anonymous andAuthenticated users. All the public methods inside the Controllers can be easily accessed if one knows the method name and the route pattern.  You can read the detailed article on Authorize attribute and its significance  [**here**](http://www.csharpstar.com/authorize-attribute-in-asp-net-mvc-and-its-significance/).  **10. What is Attribute Routing in ASP.net MVC?**  In ASP.NET MVC, we use friendly URLs that are mapped to controller’s actions instead of physical files as in case of ASP.NET WebForms. Now in ASP.NET MVC5, we can use attributes to define routes giving better control over the URIs  **11. What is Scaffolding in ASP.net MVC and what all are the advantages of using it?**   * Scaffolding is a code generation Framework * It is developed by Microsoft to perform CRUD (Create, Retrieve, Update and Delete) operations for us. * Scaffolding engine uses Microsoft’s T4 templates to generate basic controllers and views for the models * Scaffolding blends with Entity Framework and creates the instance for the mapped entity model and generates code of all CRUD Operations   **Advantages**:   * RAD approach for data-driven web applications. * Minimal effort to improve the Views. * Data Validation based on database schema. * Easily created filters for foreign key or Boolean fields     **12. What is the difference between ASPX View Engine and Razor View Engine?**   |  |  |  |  | | --- | --- | --- | --- | | **ASPX View Engine** | **Razor View Engine** | | | | **System.Web.Mvc.WebFormViewEngine** is thenamespace for ASPX View Engine. | Namespace for ASPX view Engine is **System.Web.Razor**. | | | | From the beginning, ASPX View Engine was part of ASP.NET MVC. | Razor View Engine was introduced in ASP.NET MVC v3. | | | | ASPX View Engine uses syntax same as that of Web Form pages (already demonstrated above). | Razor Syntax is different as compared to Web Forms.  Using Razor syntax, developer type comparatively less code  Which is easy to understand. | | | | ASPX syntax is inherited from Web Forms, so it’s understandable for Web Forms developer but it’s not that much clean as compared to Razor View Engine. | As Razor View Engine is introduced later in MVC3, its syntax is designed to be clean, expressive and easy to learn. | | | | ASPX View Engine does nothing to avoid Cross-Site Scripting attacks by default. | By default, Razor View Engine encodes html tags or scripts  Before its being rendered to view that avoids Cross-Site Scripting attacks. | | | | ASPX View Engine is comparatively fast. | Razor View Engine is slow as compared to ASPX View Engine. | | | | It supports design view in Visual Studio. | | It doesn’t support design view in Visual Studio. | | | No support for Test Driven Development. | | Supports Test Driven Development. | | | File Extension for this View Engine is similar to WebForm as:  1.aspx, for Views just like Web Form pages. 2. ascx, for Partial Views & Editor Template just like User Controls. 3. Master, for Layout and Master Pages just like Master Pages in Web Forms. | | | As its new and advanced View Engine, its extensions are totally different. 1. cshtml (Razor C#), For all including Views, Partial Views, Editor Template and Layout Pages. 2. vbhtml (Razor VB.NET), For all including Views, Partial Views, Editor Template and Layout Pages. |       **13. What is ASP.net Identity?**  [ASP.net Identity](http://www.csharpstar.com/wp-content/uploads/2015/11/Identity.jpg)  **ASP.NET Identity Framework** is an open source platform and can be customized according to requirement.  Through this platform it provides a user membership in the form of store or hybrid application which eradicates every security risk to web applications.ASP.NET Identity Framework 2.0 can be used with ASP.NET Web Forms, MVC, Web Pages, Web API, and SignalR.  This platform provides the major security like Two Factor Authentication, Account Lockout, and Account Confirmation etc.,  You can read the more detailed information on ASP.net Identity [**here**](http://www.csharpstar.com/asp-dotnet-identity/).  **14. What are ASP.net MVC HTML Helpers?**  HTML Helper in ASP.NET MVC is a method returning a string. The returning string is basically a HTML string that Can render a HTML tag.  HTML helper are comparatively lightweight because they don’t have view state and event model like  Web Form Controls. Along with the built in HTML helpers, we can also create our own custom helpers to fulfill our specific needs.  Standard HTML Helpers can be categorized as follows:   * URL Helpers   + HTML Links   + Image Links * HTML Form Elements   You can use HTML Helper in a view to render HTML content. In ASP.NET MVC, an HTML helper is a method returning  A string. The returning string is a HTML string that can render a HTML tag.  HTML helper doesn’t have view state and event model like Web Form Controls.  So they are comparatively lightweight, you can also create our own custom helpers to fulfill our specific needs.  In ASP.net MVC, HTML Helpers are the equivalent of ASP.net webform controls. Like webform controls, HTML helper enables you to encapsulate the rendering of HTML.  HTML helpers are comparatively lightweight because it does not have an event model and does not use view state. Using Standard HTML helper: The ASP.net MVC framework provides a standard set of helpers that you can use to render common types of HTML elements. 1. Rendering Links: The easiest way to render an HTML link in a view to use the HTML.ActionLink() helper.  It does not link to view instead creates a link to a controller action.  The HTML.ActionLink() helper supports several parameters.   * linkText – The label for the link * actionName – The action that is the target of the link * routeValues – The set of values passed to the action * controllerName – The controller that is the target of the link * htmlAttributes – The set of HTML attributes to add the link * Protocol – The protocol for the link like HTTPS, HTTP etc. * hostname – The hostname for the Link * fragment – The fragment for the link   **@Html.ActionLink(“Csharpstar Customer Profile”, “CustomerInfo”)**  Above line of code is an example of Html.ActionLink() method that renders an HTML anchor tag and linking to a Controller action “CustomerInfo” in a View as follows:  <a href=”/Site/CompanyInfo”>TechKatak Customer Profile</a> 2. Rendering Image Links: You can’t use HTML.ActionLink() helper to render an image link.You can’t pass image tag to it. You can use Url.Action() helper to generate proper link.  The Url.Action () helper supports a set of parameters similar to HTML.ActionLink() helper  **<a href=”@Url.Action(“ViewDetails”)”><img src=”../Images/ViewDetails.jpg” alt=”View Details”></a>** 3. Rendering Form Elements: There are many HTML helpers to render form elements.   * BeginForm() * CheckBox() * DropDownList() * EndForm() * Hidden() * ListBox() * Password() * RadioButton() * TextArea() * TextBox()   In order to render HTML Form elements, ASP.NET MVC provides a number of HTML helpers as follows:  **@Html.TextBox(“strCustomerName”)**renders:  <input id=”strCustomerName” name=”strCustomerName” type=”text” value=”” />  **@Html.Password(“strPassword”)**renders:  <input id=”strPassword” name=”strPassword” type=”password” />  **@Html.RadioButton(“radGender”, “Male”, true)**renders:  <input checked=”checked” id=”radGender” name=”radGender” type=”radio” value=”Male” />  **@Html.CheckBox(“chkDuesPaid”, true)**renders:  <input checked=”checked” id=”chkDuesPaid” name=”chkDuesPaid” type=”checkbox” value=”true” /> <input name=”chkDuesPaid” type=”hidden” value=”false” />  **@Html.DropDownList (“ddlLevel”, new SelectList(new [] {“1st Grade”, “2nd Grade”, “3rd Grade”}))**renders**:**  <select id=”ddlLevel” name=”ddlLevel”> <option>1st Grade</option> <option>2nd Grade</option> <option>3rd Grade</option> </select>  **15. What is Partial view in ASP.net MVC and what all are the advantages of it?**   * Partial View in ASP.NET MVC serves the same purpose as that of a User Control in ASP.NETWeb Forms. * In order to reuse a View in our MVC application (For example, placing a header, list of items or footer etc.), * Partial Views are introduced by Microsoft that basically renders inside a Parent View   **Advantages**:  using Partial View in ASP.NET MVC has following advantages:   * Enhances reusability by packaging up common website code instead of repeating the same in different pages. * Easy to maintain. Changes in future are simple to accommodate.   You can read more on Partial view [**here**](http://www.csharpstar.com/partial-view-in-asp-net-mvc/).  **16. Explain the Dependency Injection in ASP.net MVC?**   * Dependency Injection (DI) is a design pattern that takes away the responsibility of creating dependencies from a class thus resulting in a loosely coupled system * The core features of the DI container have been abstracted out to the IServiceProviderinterface and   Are available throughout the stack. Because the IServiceProvider is the same across all components of  The ASP.NET framework a single dependency can be resolved from any part of the application.   * The DI container supports just 4 modes of operation:   + **Instance** – a specific instance is given all the time. You are responsible for its initial creation.   + **Transient** – a new instance is created every time.   + **Singleton** – a single instance is created and it acts like a singleton.   + **Scoped** – a single instance is created inside the current scope. It is equivalent to Singleton in the   Current scope.  **Advantages of Dependency Injection:**   * Reduces class coupling * Increases code reusing * Improves code maintainability * Improves application testing   You can read more detailed explanation with examples on Dependency Injection [**here**](http://www.csharpstar.com/dependency-injection-in-asp-net-mvc/).      **17. Explain the Difference between MVC3, MVC4, MVC5 and MVC6?**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | ASP.net MVC3 | | ASP.net MVC4 | ASP.net MVC5 | | ASP.net MVC6 | | | New Project Templates having support for HTML 5 and CSS 3. | ASP.NET Web API, a framework that simplifies the creation of HTTP services and serving a wide range of clients. | | ASP.NET Identity for authentication and identity management. These days, modern applications are developed for broader range of clients such as web, mobile in mind. Also, users are actively using their social identities from various social channels like facebook, youtube, twitter etc. ASP.NET Identity is a new Membership system to handle authentication and authorization for variety of clients as well as using user’s existing social identities. | | Single Programming Model for ASP.NET MVC and ASP.NET Web API. | | | Improved Model validation. | | Adaptive rendering and other look-n-feel improvements to Default Project Templates. | | Authentication Filters for authenticating user by custom or third-party authentication provider. | | Optimized for Cloud Computing. | | Razor View Engine introduced with a bundle of new features. | | Empty Project Template. | | With the help of Filter overrides, we can now override filters on a method or controller. | | Supporting side by side deployment of runtime and framework along withapplication. | | Support for Multiple View Engines i.e. Web Forms view engine, Razor or open source. | | yes | | Bootstrap replaced the default MVC template. | | Out of the box support for dependency injection. | | Controller improvements like ViewBag dynamic property and ActionResults Types etc. | | Support for adding controller to other project folders also. | | Attribute Routing is now integrated into MVC5. Basically, MVC Routing is an excellent way to create human friendly and Search Engine Optimized URLs. You can easily get understanding about Routing in ASP.NET MVC here. Attribute based routing enables us to define routes along with action methods as follows: | | vNext is Open Source and supports running on multiple platforms including Linux and Mac. | | Unobtrusive JavaScript approach that separates the functionality from presentation layer on a web page. | | Task Support for Asynchronous Controllers. | |  | | New JSON-based project Extension. | | Improved Dependency Injection with new IDependencyResolver. | | Controlling Bundling and Minification through web.config. | |  | | In order to dynamically compile code, Roslyn compiler is used. | | Partial page output caching. | | Support for OAuth and OpenID logins using DotNetOpenAuth library. | |  | |  | |  | | Support for Windows Azure SDK 1.6 andnew releases. | |  | |  |   **18. What is ViewEngine in ASP.net MVC?**  View Engine in ASP.NET MVC is used to translate our views to HTML and then render to response.  Multiple View Engines are available for MVC including ASPX, Razor, NHaml etc.  Normally in ASP.NET MVC, a View Engine translates view into HTML by:   * Providing implementation of **IViewEngine** (as template provider) * **IView** (as rendering template) and * A **Template Engine** for parsing and compiling view file into executable code.   ASPX View Engine is also known as Web Form View Engine. it’s syntax as “<%= %>” or “<%: %>” for rendering  Server-side contents.  **19. What is Bootstrap in MVC5?**   * Bootstrap (a front-end framework) is an open source framework that makes use of HTML,CSS,JavaScript for   building responsive web applications.   * Bootstrap provides a base collection including layouts, base CSS, JavaScript widgets, customizable   components and plugins.   * Bootstrap is developed by 2 developers from twitter to accelerate web development   You can read more detailed article on Bootstrap [**here**](http://www.csharpstar.com/bootstrap-overview-in-asp-net-mvc/).  **20. What is ASP.net vNext?**  [ASP.net vNext](http://www.csharpstar.com/wp-content/uploads/2015/11/vnext.jpg)  **ASP.NET vNеxt** соnѕidеrѕ the ASP.net features from very begining till рrеѕеnt dау.It supports:   * **Bеttеr execution** * **Lоwеr аѕѕеt utilizаtiоn** * An **аѕуnс mоdеl** for еnhаnсеd еxесutiоn * **Adaptability** * Eсоnоmу in **cloud-based fасilitаting ѕituаtiоnѕ**.  In this article, we will learn:  * What is Garbage Collection? * How the Garbage Collector works? * Destructors and Dispose Method * Manage unmanaged resources * Implementing IDisposable and Finalizer * How to Force Garbage Collection * Implementing Dispose method * Difference between Dispose and Finalize methods * Using Weak References * Summary  What is Garbage Collection? When a program starts, the system allocates some memory for the program to get executed.  When a C# program instantiates a class, it creates an object.  The program manipulates the object, and at some point the object may no longer be needed. When the object is no longer accessible to the program and becomes a candidate for garbage collection.  There are two places in memory where the CLR stores items while your code executes.   * stack * heap   The stack keeps track of what’s executing in your code (like your local variables), and the heap keeps track of  your objects. Value types can be stored on both the stack and the heap.  For an object on the heap, there is always a reference on the stack that points to it.  The garbage collector starts cleaning up only when there is not enough room on the heap to construct a new object The stack is automatically cleared at the end of a method. The CLR takes care of this and you don’t have to worry about it.  The heap is managed by the garbage collector.  In unmanaged environments without a garbage collector, you have to keep track of which objects were allocated  On the heap and you need to free them explicitly. In the .NET Framework, this is done by the garbage collector. How the Garbage Collector works? Let’s look at below diagram to understand it better. [Garbage Collection](http://www.csharpstar.com/wp-content/uploads/2016/03/Garbage_Collection.jpg) Before Garbage Collector Runs: In the above diagram, Before Garbage collector runs, the application root has dependency on object 1, object 3 and object 5. Object 1 is dependent on object 2 and Object 5 is dependent on object 6. So the application root does not have any dependency on object 4 and object7. When Garbage collector runs:Marking Phase:  * It marks all the heap memory as not in use * Then examines all the programs reference variables, parameters that has object reference, CPU registers and other items that point to heap objects  Relocating Phase:  * For each references, the garbage collector marks the object to which the reference points as in use  Compact Phase:  * Then it compacts heap memory that is still in use and updates program reference * Garbage collector updates the heap itself so that the program can allocate memory from unused portion  After the Garbage collector runs: It discards Object 4 and Object 7 since there is no dependency exists and compact the heap memory.  When it destroys an object, the garbage collector frees the object’s memory and any unmanaged resource it contains.  You can use Destructor and the Dispose method to determine, when and how the object frees managed and unmanaged resources. Destructors:  * Destructors can be defined in classes only, not structures. * A class can have at most one destructor. * Destructors cannot be inherited or overloaded. * Destructors cannot be called directly. * Destructors cannot have modifiers or parameters.  Destructor to Finalizer: The GC actually calls an object’s finalizer, not its destructor. The destructor is converted into an override version of the Finalize method that executes the destructor’s code and then calls the base class’s Finalize method.  For example, suppose the Employee class includes the following destructor:   |  |  | | --- | --- | |  | ~Employee()  {  // Free unmanaged resources here.  ...  }  This destructor is converted into the following Finalize method:  protectedoverride voidFinalize()  {  try  {  // Free unmanaged resources here.  ...  }  finally  {  base.Finalize();  }  } |   You cannot explicitly override the Finalize method in C# code. Dispose Method: C# defines the IDisposable interface, which declares the Dispose method. If a class implements this interface, then the using statement will automatically call an object’s Dispose method, so you don’t need to do it explicitly.  If the Dispose method has freed all object’s resources then no need to invoke destructor.  Dispose method can call GC.SuppressFinalize to tell the Garbage Collector to skip Object’s Destructor.  Let’s look at below example to see the implementation of Dispose method in C#.   |  |  | | --- | --- | |  | **Public class MyWrappedResource:IDisposable**  **{**  **//our managed resource**  **IDbConnection \_conn=null;**  **publicMyWrappedResource(stringfilename)**  **{**  **}**  **publicvoidClose()**  **{**  **Dispose(true);**  **}**  **publicvoidDispose()**  **{**  **Dispose(true);**  **}**  **privatebool\_disposed=false;**  **protectedvoidDispose(booldisposing)**  **{**  **//in a class hierarchy, don’t forget to call the base class!**  **//base.Dispose(disposing);**  **if(!\_disposed)**  **{**  **\_disposed=true;**  **if(disposing)**  **{**  **//cleanup managed resources**  **if(\_conn!=null)**  **{**  **\_conn.Dispose();**  **}**  **}**  **//cleanup unmanaged resources here, if any**  **}**  **}**  **}** |    Rules on Resource management:  * If a class contains no managed resources and no unmanaged resources, it doesn’t need to implement IDisposable or have a destructor. * If the class has only managed resources, it should implement IDisposable but it doesn’t need a destructor. * If the class has only unmanaged resources, it needs to implement IDisposable and needs a destructor in case the program doesn’t call Dispose. * The Dispose method must be safe to run more than once. You can achieve that by using a variable to keep track of whether it has been run before. * The Dispose method should free both managed and unmanaged resources. * The destructor should free only unmanaged resources. * After freeing resources, the destructor should call GC.SuppressFinalize, so the object can skip the finalization queue.    Managing Unmanaged Resource: Garbage collector will be taking care of managed resources.But when you will deal with Unmanaged resources like network connection, file handle, window handle etc.. You have to explicitly release those items. If not, you will get errors such as “This File is in use” or you won’t be able to connect to your database because all connections are in use.  To handle unmanaged resources, C# supports the concept of finalization. This mechanism allows a type to clean up prior to garbage collection. But in C#, you can’t be sure when a finalizer is called. It will happen only when the garbage collector determines that your object is ready for being cleaned up. A finalizer in C# requires some special syntax, just as a constructor. You need to prefix the class name with a tilde (~) to create a finalizer. ExampleAdding Finalizer:  |  |  | | --- | --- | |  | Public class Finalizer  {  ~FinalizerExample()  {  // This code is called when the finalize method executes  }  } |   Inside the finalizer, you can clean up other resources and make sure that all memory is freed. Note: The finalizer is called only when a garbage collection occurs. Force Garbage Collection: You can force this by adding a call to GC.Collect. Example  |  |  | | --- | --- | |  | StreamWriter stream=File.CreateText(“temp.dat”);  stream.Write(“some test data”);  GC.Collect();  GC.WaitForPendingFinalizers();  File.Delete(“temp.dat”); |   The line WaitForPendingFinalizers makes sure that all finalizers have run before the code continues. It is not recommended that you call GC.Collect yourself.  A finalizer increases the life of an object. Because the finalization code also has to run, the .NET Framework keeps a reference to the object in a special finalization queue. An additional thread runs all the finalizers at a time deemed appropriate based on the execution context. This delays garbage collection for types that have a finalizer.  Implementing IDisposable and Finalizer:  |  |  | | --- | --- | |  | using System;  using System.IO;  classUnmangedWrapper:IDisposable  {  publicFileStreamStream{get;privateset;}      publicUnmangedWrapper()          {          this.Stream=File.Open(“temp.dat”,FileMode.Create);          }      ~UnmangedWrapper()          {          Dispose(false);          }  publicvoidClose()      {          Dispose();      }  publicvoidDispose()      {          Dispose(true);          System.GC.SuppressFinalize(this);      }  publicvoidDispose(booldisposing)      {          if(disposing)              {              if(Stream!=null)                  {                  Stream.Close();                  }          }      }  } |  Difference between Dispose and Finalize:  |  |  | | --- | --- | | Dispose | Finalize | | It is used to free unmanaged resources at any time. | It can be used to free unmanaged resources held by  an object before that object is destroyed. | | It is called by user code and the class which is implementing dispose method, must has to implement IDisposable interface. | It is called by Garbage Collector and cannot be called  by user code. | | It is implemented by implementing IDisposable interface Dispose() method. | It is implemented with the help of Destructors | | There is no performance costs associated with Dispose method. | There is performance costs associated with Finalize method  since it doesn’t clean the memory immediately and called  by GC automatically. |  [Difference between == and .Equals method in c#](http://www.csharpstar.com/difference-between-and-equals-method-in-csharp/) November 18, 2015 [0](http://www.csharpstar.com/difference-between-and-equals-method-in-csharp/#respond) For Value Type: **==** and .**Equals**() method usually compare two objects by value.  For Example:  int x = 20;  int y = 20;  Console.WriteLine( x == y);  Console.WriteLine(x.Equals(y));  Output:  **True**  **True** For Reference Type: **==** performs an identity comparison, i.e. it will only return true if both references point to the same object. While **Equals**() method is expected to perform a value comparison, i.e. it will return true if the references point to objects that are equivalent.  For Example: StringBuilder s1 = new StringBuilder(“Yes”);  StringBuilder s2 = new StringBuilder(“Yes”);  Console.WriteLine(s1 == s2);  Console.WriteLine(s1.Equals(s2));  Output:  **False**  **True**  In above example, s1 and s2 are different objects hence “==” returns false, but they are equivalent hence “Equals()” method returns true. Remember there is an exception of this rule, i.e. when you use “==” operator with string class it compares value rather than identity.  **When to use “==” operator and when to use “.Equals()” method?**  For value comparison, with Value Type use “==” operator and use “Equals()” method while performing value comparison with Reference Type. [Is String a Value Type or a Reference Type in C# ?](http://www.csharpstar.com/string-value-type-reference-type-csharp/) August 31, 2016 [2](http://www.csharpstar.com/string-value-type-reference-type-csharp/#comments) In this article, we will discuss: – What is Stack & Heap memory? – What is value type & Reference type in C#? – Is string a value type or Reference type?  What is Stack & Heap memory? The stack is an array of memory. The heap is an area of memory where chunks are allocated to store certain kinds of data objects. Unlike the stack, data can be stored and removed from the heap in any order. Your program can store items in the heap, it cannot explicitly delete them. Instead, the CLR’s garbage collector (GC) automatically cleans up orphaned heap objects when it determines that your code can no longer access them.  You can read more on stack vs Heap memory [here](http://www.csharpstar.com/difference-between-stack-and-heap-memory-csharp/). What is value type & Reference type in C#? There are 3 different categories of Types in C#. – Value Type – Reference Type – Pointer Types  Pointer types are rarely used. You use them only when working with unsafe code and when you need to use pointer arithmetic.  A Value Type holds the data within its own memory allocation and a Reference Type contains a pointer to another memory location that holds the real data.  You can read the difference between Value type and reference type [here](http://www.csharpstar.com/value-type-vs-reference-type-in-csharp/). Is string a value type or Reference type?This is a frequently asked interview question. Every developer should know String is a reference type and behaves like value type. In .Net Framework Strings are immutable reference types. All .net datatypes has default size except string and user type. So String is a Reference type, because it does not have default allocation size.  Immutable means, it cannot be changed after it has been created. Every change to a string will create a new string. This is why all of the String manipulation methods return a string.  For an example, an integer (System.Int32) has a fixed memory size (4 bytes) of Value range -2,147,483,648 through 2,147,483,647. Hence, an integer can be stored on the Stack (i.e. fixed memory). Alternatively, a String does not have a pre-defined memory size and it can be huge (the value range may be 0 to approximately 2 billion Unicode characters), so it requires dynamic memory allocation.  When a String object is created, the actual value is stored within dynamic memory, or on the Heap.  Reference types have some overhead on construction and destruction and garbage collection, because they are created on the heap. Value types on the other hand have overhead on method calls (if the data size is larger than a pointer), because the whole object is copied rather than just a pointer. Because strings can be much larger than the size of a pointer, they are designed as reference types. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1. What is Framework?** Framework is special kind of libraries or structures that helps in building applications.   **2. What is .Net Framework?** It is collection of libraries and API designed by Microsoft which helps in building, deploying and running applications and services that use >net technologies. In simple words a .net framework is a combination of several diff technology like CLR, asp.net, garbage collection etc. that helps in easing the process of software development.    **3. What are the features of .Net framework?** MSIL,CLR,CTS,Assemblies   **4. Briefly explain how the code is compiled by the framework?** The .net code first is compiled into MSIL/IL i.e intermediate language which is machine independent. At this stage metadata is generated which contains the member signature. This code then gets compiled into machine specific language known as native languageby JIT (just in time) compiler.  **5. What is JIT?** Just -In-Time compiler compiles the IL code to native code so that the machine can understand that language. JIT runs on demand when a method is called JIT analysis the IL and produces highly efficient code. So if the next time same piece of code i execitued then it runs and not compiled. JIT is specific to OS**.**  **6. What is AOT?** Ahead-Of-Time compilation. The JIT runs just before the code execution hence it does not have enough time of optimize the code. Hence cam AOT. It compiles the entire .net assembly into IL during application installation. ngen is a type of AOT  **7.What is ngen.exe ?** Native Image Generator. It compiles the entire assembly during installation. It creates native images that are nothing but files so that the next time compiles compiles it picks up these images instead of files from JIT. This helps in faster compilation. It can be used only if ngen is used for all the assembly in the application.  **8. How MSIL helps in language interoperability?** MSIL is machine independent language.no matter which language you use in .net all of then are converted into MSIL .thus helping diff language coexist with each other.   **9. What is CLR?** Common language runtime.it is the heart of .net framework and it takes care of code execution. It helps in running your program by acting as a translator between your program and the computer you are running it on. Features of CLR - a) *Garbage Collection*-helps in automatic memory management by disposing or the objects when they are not being referenced anymore.  b) *IL to native code* -CLR uses JIT compiler to convert MSIL to native code. c) Code access security (CAS)-provides right to certain programs. Example, if a piece of code is trying to delete a file for which it does not have permission, CLR prevents it from doing so. d) Code verification-ensures type safety and prevents code from accessing invalid memory locations etc.   **10. What is CTS?** Common type system, defines set of data types that are common for all the languages in .net.eg-long in vb.net and int in c# both point to system.int32 in IL.  **11. What is CLS?**  Common language specification**,** defines rules that all the languages must follow in order to co-exist among each other.  **8. What are languages supported in .Net?** It supports over 60 languages some of them are-c#,ada,C++, j#,Perl,VB,cobol etc.  **9. What is .Vshost ?**  When we build our vs project this exe is created in the bin folder.it is a hosting process. And is used only by visual studio.it helps in improving performance and supports immediate window.  **10. What is managed and unmanaged code?** The code which runs under CLR is called managed code eg-VB, C#. the code which is not compiled by CLR is called unmanaged code.eg-C++ [WCF](http://csharptopicwiseques.blogspot.in/2010/08/wcf.html) **1. What is WCF?**  Windows Communication Foundation is a framework for building Service Oriented Application (SOA).It was introduced in .Net Framework 3.0 and is basically a combined feature of WebService, Remoting, MSMQ and COM+.  **2. Difference between Webservice and WCF ?**  ->WebService cab be  hosted only in IIS  WCF can be hosted in IIS, windows Service, Self-hosting and windows activation service.  ->WebService can be accessed only through http protocol. WCF cab be accessed through http,tcp,MSMQ and named pipes.  ->WebService works in stateless env. WCF can maintain states and sessions**.**  ->Webservice uses System.Xml.serialization name space for serialization. WCF uses System.Runtime.Serialization namespace  for serialization  **3. What is .svc file?**  It’s the file where service is defined and it will be the point of contact from  the consumers.It contains name of service and code behind file name. It is used to know about the service.  **4. What are different ways in which WCF can be hosted?**  The different hosting techniques are 1.IIS hosting--This is the most commonly used hosting technique. Here IIS is used as a server and has following  advantages -> starts automatically on the first client request, process recycling.  Disadvantage-- It supports only http protocol  2.Self hosting-This is hosting the WCF  urself either in console/windows  application also  in windows service .   * The host process should be running before a client makes a call to servc. * It’s easier to debug and deploy. * Lifetime of services can be controlled using Open and Close methods.   3. WAS hosting(Windows Activation Server hosting)--WAS was introduced with windows  vista and it is shipped with IIS 7.0. it is more powerful than IIS 6.0 as it can support  http, tcp and named pipes whereas IIS 6.0 can  support only http.   4.Windows Service Hosting--Here the service can be programmed to start  when the system starts.    **5. What do you mean by ABC in WCF ?**  ABC in WCF means - Address -> A stands for Address. It indicates where you service is located. A URL is used to point to the location. Depending on whether the service is hosted in http, tcp... the address varies. e.g.- http://localhost/Test net.tcp//localhost/Test  Binding ->B stands for Binding which specifies how the client should communicate with the service.  Contract->C stands for contracts. It exposes the operation provided by the Service. It is a contract.    **6. What is an endpoint in WCF ?**  ABC together constitute an endpoint. Endpoint provides the client access to the functionality exposed by the service.     **7. What do you mean by contracts. What are the different types of contracts?** Contracts are standard way of describing the operations performed by a Service. Diff types of contracts are--   1.Service Contracts  It describes the operations provided by the service.  WCF service should have at least one ServiceContract.  It has two attributes- [ServiceContract] --Used to define the interface. Client can talk to services via interfaces. Similar to [Webservice] attribute in Webservice  [OperationContract]-Used to define methods inside the interface. Similar to  [WebMethods]  **eg-**                   [ServiceContract()]                   public interface ISimpleCalculator                  {                      [OperationContract()]                      int Add(int a, int b);                   }   1. DataContract    This defines the data which is exchanged between the client and the service.   These are used mainly for used defined objects.  They are of two types   [DataContract]--defines the class   [DataMember] --defines the properties.  eg-  [DataContract]  class Name  {  public string Fullname;  [DataMember]  public string FirstName;  [DataMember]  public string SurName;  }  Inbuilt types like string (Fullname) are defined impicitly and they do not need  data contracts.     1. Fault Contracts.     These are used to handle errors in a service. When a service throws an error it does not reach the client side.  But by using fault contract we can come to know the error that is raised by the service. eg-  [ServiceContract()]  public interface ISimpleCalculator  {  [OperationContract()]  [FaultContract(typeof(ArgumentException))]  int Add(int a, int b);  }  Raise the exception as  throw new FaultException("error") ;  4. *Message Contracts*  Message Contract defines the way messages are transferred using SOAP messages. It is used if you want to customize the SOAP format. eg-  [MessageContract]  public class CustomerDetails  {  [MessageHeader]  public string CustID;  [MessageBodyMember]  public string Name;  }    **8. What’s the diff between Message Contract and Data Contract.** Data contracts are used to describe data types used by service. They can be either parameter or return type. Message Contract are used to describe SOAP message format. They allow us to control the detail is SOAP header and body.  **9. What namespace is used to access WCF?** System.ServiceModel  **10.What are the main components of WCF?** 1. Service Class--functional class using any .net language. 2. Hosting environment--IIS, windows service etc. 3.Endpoint--ABC  **11.what is proxy in WCF?**     A proxy is a class by which a service client can Interact with the service. By the use of proxy in the client application we can call different methods in the service.   **12. What are two method in which you can access WCF?** By Adding it as reference -- this will automatically generate the channel required for you to communicate with WCF. By creating channel manually- use ChannelFactory. however the second method should be preferred one as pointed out in below article <http://www.eggheadcafe.com/tutorials/aspnet/a1647f10-9aa4-4b0c-bbd9-dfa51a9fab8e/adding-wcf-service-refere.aspx>    **13. How do you achieve method overloading in WCF?**  By making use of name property in OperationContract  [ServiceContract] interface ISimpleCalculator {  [OperationContract(Name="Add2numbers")]  int Add(int a,int b)   [OperationContract(Name="AddDouble")]  int Add(double a,double b) }  **14. Why normal method overloading is not possible in WCF ?**  Because WSDL is not a OO language and it does not support OOPs concept. [Exception](http://csharptopicwiseques.blogspot.in/2010/09/exception.html) **1. What are the two diff types of error?**  Run time error and Compile time error  **2. What is the base class from which all the exceptions are derived?**  System.Exception class  **3. What are the two main kind of exceptions class?**  *System Exception*-- Mainly used for inbuilt exceptions like Null Ref Exception,Overflow Exception,Argument ....etc  *Application Exception*-This class is used for user defined exceptions.  **4. How to you handle error in C#?** By using try, catch and finally blocks. *try*-contains the code which might throw an exception *catch*-deals with various error conditions *finally*-does the cleaning up.  **5. Does finally get executed if the code throws an error?** Finally is always executed.  **6. Is it mandatory for a piece of code to have catch or finally when try block is there?** Yes. If a try block is there then either catch or finally has to be there or else a compiler error is generated.  **7. Can finally bock have return statement?** No, a compile time error is generated. That’s bc finally contains the cleanup code.  **8. Can there be many catch statements for a single try block?** Yes. But the rule is most specific exceptions should be caught first.  **9. Will the foll code compile?**  try              {                 throw new SystemException();             }                         catch(Exception e)             {             }             catch(ArgumentException e)             {             } No.It gives the foll error A previous catch clause already catches all exceptions of this or of a super type ('System.Exception')  **10.Whats the syntax for catching all kinds of exception.**  Try  {  } catch         //no parameters specified { } it will catch exceptions thrown by managed as well as unmanaged code**.**  **11. All the foll catch block uses the same exception parameter 'e'. Is it valid?** catch(Exception e) {} catch(ArgumentException e) {}  Yes,bc the parameter passed to catch block wil be within its scope only.  **12. What are diff exception properties?**  a)Helplink-- provide link to help files which give more info on the exception raised.  b)*Message*--gives the text that describes the exception. Suppose catch(Exception e)                 {                        throw new Exception("Failure")                 } then e.Message will give output as "Failure".  c)*Source*--provides the name of assembly from where exception is raised. d)*Target Site*--name of method thats trows an error e)*Stack Trace*--provides stack flow of the exception. f)*Inner Exception*--When exception is thrown from one catch to another the message from first catch is passed as inner exception to second catch.  **13. What is checked and unchecked operator?** *checked*--checks  for the error and throws an exception. *unchecked*--does not check for the error .truncates the data and prints it is used mainly while casting.  **14. Does following code work?** try {} catch(Exception e) {} catch(Exception e) {}  No,two catch block cannot have same exception classes.  **15. Whats the diff bt throw and throw e ?** *throw*-- throws the exception.                throw is internally compiled to rethrow   *throw e*--throws the exception and resets the stack trace to current catch                   location.                   Internally compiled to throw which taken an exception parameter as                   input.  **16. What is the output of foll programs**    a)       try              {                 throw new SystemException("This is wrong");             }             catch(Exception e)             {                 Console.WriteLine(e.Message);                              return;             }             finally             {                 Console.WriteLine("i am inside finally");                 Console.Read();             }  **output**--   This is wrong                      i am inside finally.  b)        try              {                 throw new SystemException("This is wrong");             }             catch(Exception e)             {                 Console.WriteLine(e.Message);                 throw;             }             finally             {                 Console.WriteLine("i am inside finally");                 Console.Read();             }  **output**- This is wrong.                    Application crashes bc the throw is not being caught anywhere and                   hence finally won’t be executed.  c)public class A          {         public void methodA()          {             try              {                 B test = new B();                 test.methodB();             }              catch (Exception e)                         {                 throw e;                 Console.WriteLine("Exception from Class A");             }             finally                        {                 Console.WriteLine("Inside Class A Finally");             }         }     }      public class B      {         public void methodB()          {             try              {                 throw new Exception("Exception from Class B");             }              catch (Exception e)              {                 throw;                 Console.WriteLine("Exception from Class A");             }              finally              {                 Console.WriteLine("Inside Class B Finally");             }         }     }  output--inside class B finally.   **Not able to see the line number in exception stack trace in remote machine**  While building a project make sure .pdb files are generated. copy these files in the folder where you have installed your application in remote machine . If pdb files are not generated check if the setting is enabled for your project-  Solution->Properties->Build->Advanced Build Settings->Debug Info->pdb-only   [Value Type and Reference Types](http://csharptopicwiseques.blogspot.in/2010/08/value-type-and-reference-types.html) **1. What are Value Types? Give example** Variables that store data are called value types. Value types are stored on stack. They contain the actual values. e.g. - int, enum, structs.  **2. What are Reference Types? Give Example.** Variables that store reference to actual data are called Reference types. Reference types stored on heap but contain the address on heap. eg-class, interface, delegate, string, object**,**Array  **3. Difference between value type and reference types?**   |  |  | | --- | --- | | **Value Type** | **Reference Type** | | They are stored on stack | They are stored on heap | | Contains actual value | Contains reference to a value | | Cannot contain null values. However this can be achieved by nullable types | Can contain null values. | | Value type is popped on its own from stack when they go out of scope. | Required garbage collector to free memory. | | Memory is allocated at compile time | Memory is allocated at run time |   **4.Difference between stack and heap**   |  |  | | --- | --- | | **Stack** | **Heap** | | Values are stored on one another like a stack. | Values are stored in random order.like dumped into a huge space | | Used for value type | Used for reference types | |  |  |   **5. Whats the output of the following program?** 1.       int x = new int();                 x = 20;                 int y = new int();                 y = x;                         y = 30;     //Line3                     return x;              Output--20             At Line 3 only y value is changes. "x" has its own location on stack which is not              affected by this and hence retains is old values.        2.       class Numbers                 {                    public int MyValue;                }                              Numbers x = new Numbers();                x.MyValue = 3;                Numbers y = new Numbers();                y = x;                y.MyValue = 4;                Console.WriteLine( x.MyValue);                             Output- 4. Both x and y are pointing to the same location on the heap.          **6. What is boxing and unboxing?**  Boxing-- Converting a value type to a reference type .This is implicit.                      During boxing following 3 things happen-                      Memory is allocated on heap,                      Value is copied from stack to heap,                       Reference is updated to point t heap **e.g.-**  int i = 123;  object o = i;  here if you make i= 456;  then o=123, //value does not change    Unboxing- Converting reference type to value type.  The value which is being unboxed has to be boxed first.  eg-int j = (int) o;      **7.Are Value Type sealed ?**  Yes  **8.Can you implement IDisposable for value type ?** Structs being a value type can inherit IDisposable interface  **10.how do you get size of value type ?** using SizeOf operator.   **11.When you declare int i,j hos is it alocated?** Both of them are allocated on the stack at the same time and they go out of scope at the  same time**.**  **12. What do you mean by Pass By Value?** Actual value is copied to another variable  eg-               calling method--                    Actual Method                      int a=4,                                    Add(int a,int b){               int b=5,                                     a=6,b=7 }              Add(4,5)              Print a and b --Line 1 Even though the value of a and b is changed in the actualfunction it does not  chane in  line 1.  **13. What do you mean by Pass By Reference?** The calling functions passes the address of the variable instead of the actual values.Any  changes made inside the actual method will be reflected in the calling function.  e.g.-               calling method--                    Actual Method                      int a=4,                                    Add(ref int a,ref int b){               int b=5,                                     a=6,b=7 }              Add(ref 4,ref 5)              Print a and b --Line 1  The value at Line 1 will be a=6,b=7       **13.What do you mean by Pass By out?** This is similar to pass by ref. Only diff is before passing the values they need not be  initialized. But they have to be initialized inside actual method. They are mainly used in remoting. eg-                calling method--                    Actual Method                      int a,                                       Add(out int a,out int b){               int b,                                       a=6,b=7 }              Add(out 4,out5)              Print a and b --Line 1  The value at Line 1 willl be a=6,b=7      **14.what are params ?**  They are used when you do not know how many variables you will be passing across methods. **eg-**            calling method--                     Actual Method                   method(1,2,3,4)                      method(int a,params int[])    The param array should be last while declaring method arguments.        **15. What the difficult parameter type used when calling methods in c#?** Pass by Value  **13. What are diff types of parameter in c#?** Value parameter(in), Reference/ inout paramete  ref),Output parameter(out),   Parameter  Array(param)     [SQL](http://csharptopicwiseques.blogspot.in/2010/09/sql.html) **1. How do you create a table in SQl?** Create table <table name> (<column name1><datatype>, <column name2><datatype>) Create table A (Name varchar(20),Age int)   **2. How do you add new column to a table?** Alter table <tablename> add <columnname><datatype>  Alter table A Add Location varchar(20).  **3. How do you remove a column from a table?** Alter table <tablename> drop  coulmn <columnname> Alter table A drop column Location.  **4. Diff between Drop , Delete and Truncate.**   |  |  |  | | --- | --- | --- | | **Drop** | **Trucate** | **Delete** | | DDL Commands | DDL commands | DML command | | No Rollback possible | No Rollback possible | Can be rolled back | | Drops the complete table along with structure and data | Removes the data | Removes the data | | Drops complete table | Drops complete table | Selective deletion is possible |   **5. Select all the students whose age is between 10 to 15.**  Student(Name,Age)  select name from Student where Age between 10 and 15  **6.Diff bt Primary Key and Unique Key.** Both are used to uniquely identify rows  Primary Key-Cannot be null,Only one primary key can exist in a table Unique Key-can be null**,**can have more than one unique key in table**.**  **7. Whats the diff bt having and where clause?** Where -used in all DML statements.used only with group by function Having -used with aggregate or group.can be used only with SELECT   **8.How to you copy large data to SQL Server?** Using a tool called Bulk Copy.  **9. Find all the students whose marks is greater than the average marks.** Student(Name,Marks) Select Name from Student having Marks>Avg(Marks) group by Name.  **10. What’s the difference between group by and orderby ?** Group by-This is used only with aggregate functions. It sorts by rows. The results are displayed in groups. Orderby-mainly used with select statements. Sorts by columns.  **11. What is a default TCP/IP socket assigned for SQL Server?** 1433  **12.Whats the output of the foll sql statement-Student(Name,Roll no) contains 3 rows.Select 1,2,'India' from Student**  It prints 3 columns and 3 rows. The first rows all the column contains 1. The second column contains 2 and 3rd contains India.  **13. How do you delete repeated data by maintaining at least on occurrence of the data?**  Student(Name,Roll no)  A) If there are 2 repeated records it deletes the one occurrence.  delete top(1) from student where name='Cha'  b)select distinct \*  from student  c) Row count can be set. The below queries deletes only one occurrence of repeated data.  set rowcount 1  delete from student where name='cha'  **Stored Procedure**  **1. What are Stored Procedure?** They are set of SQL statements which have been grouped together and stored in database.  They can be executed over and over again.  **2. How to compile stored procedure on each execution?**  By using recompile option.  exec sp\_recompile MyStoredProcedure.  **3. Write a SP using input parameters.**  create procedure sp  @id int as select \* from <tablename> where ID=@id  exec sp 22   **4.Write a SP to using input and output parameters.**  create procedure sp  @id int, @Name varchar output as select Name from recent where ID=@id  declare @Name varchar exec sp 22,@Name output  You have to declare the output parameter before printing it.  **5. Write a sp to update a column taking user inputs.**  create procedure sp  @Name varchar(20), @id int as update  <tablename> set Name =@Name where ID=@id  exec sp 'test' ,22  **6. How Implement error handling in stored procedure?**  There are many diff ways- a)Try-Catch Block b)@@Errorcount  **7.Whats the difference between user defined function and stored procedure.**  A user-defined function is a routine that encapsulates useful logic for use in other queries  a) UDF functions can be used with select statement.Sp cannot. b) SP can /cannot return more than one value.UDF has to return values and it can return only one value. c) SP can have both input and output parameters wheras UDF can have only input parameters. d) Sp can have try catch block to handle exceptions, this cannot be done in UDF. e) UDF can be called from procedure whereas vice versa is not possible.  **Trigger**  **1. What is Trigger?**  They are special kind of stored procedure that are automatically executed. They are executed when an insert,update or delete operation is performed.  **2. Can you fire Trigger from c#?**  No  **3. Can stored procedure fire Triggers?**  Yes  **JOINS**      Student                     Teacher  [http://2.bp.blogspot.com/_PNI2NmTKsfQ/TISjYVPSkgI/AAAAAAAAA8I/syiAmhTcGvk/s320/1.jpg](http://2.bp.blogspot.com/_PNI2NmTKsfQ/TISjYVPSkgI/AAAAAAAAA8I/syiAmhTcGvk/s1600/1.jpg)  **1. What do you mean by cross join ?** It joins both the tables and returns the Cartesian product of both of them **.**  *select \* from  Student,Teacher*  [http://3.bp.blogspot.com/_PNI2NmTKsfQ/TISj8-yZrtI/AAAAAAAAA8Q/0B0a2Ap3bug/s320/1.jpg](http://3.bp.blogspot.com/_PNI2NmTKsfQ/TISj8-yZrtI/AAAAAAAAA8Q/0B0a2Ap3bug/s1600/1.jpg)  **2. What is inner join?** It joins two tables based on the at least one match in both the tables. select \* from Student s inner join teacher t on s.id=t.id  **3.what is left outer join ?** Prints all the values in the first table even if there are no matches in the second table .corresponding unmatched value in second table is printed as null  **3.what is right outer join?** Prints all the values in the second table. corresponding value in first table is printed as null  **UNION**  **1. What is union?** Combines only distinct values from both the tables into a single table. *select \* from Student union  select \* from Teacher*  [http://3.bp.blogspot.com/_PNI2NmTKsfQ/TISkZLIwwOI/AAAAAAAAA8g/1BCpHPncexw/s320/1.jpg](http://3.bp.blogspot.com/_PNI2NmTKsfQ/TISkZLIwwOI/AAAAAAAAA8g/1BCpHPncexw/s1600/1.jpg)  *select \* from Teacher union  select \* from Student*  [http://2.bp.blogspot.com/_PNI2NmTKsfQ/TISl7oL9uGI/AAAAAAAAA8w/bIVVXOuUsvY/s320/1.jpg](http://2.bp.blogspot.com/_PNI2NmTKsfQ/TISl7oL9uGI/AAAAAAAAA8w/bIVVXOuUsvY/s1600/1.jpg)  **2. What is union all?** Combines all the values from both the tables.   *select \* from Teacher union all select \* from Student*  [http://4.bp.blogspot.com/_PNI2NmTKsfQ/TISljDnVGBI/AAAAAAAAA8o/HDu-J0wS3wo/s320/1.jpg](http://4.bp.blogspot.com/_PNI2NmTKsfQ/TISljDnVGBI/AAAAAAAAA8o/HDu-J0wS3wo/s1600/1.jpg)  **View** **1.What is View?** It is a virtual table which encapsulated many complex queries.  **2. Advantages and disadvantage of using a view?** a) A temp table can be created by joining many tables. b) selective exposure of data from the original table.  c) restrict the access to the original table.  d) They do not consume space as they are created dynamically. *Disadvantage* a) When  table is dropped view becomes inactive b)cannot use DML if view is created using more than one table  **3.How do you create a View?** Create View<Viewname> as select \* from <tablename>  **4. If data is changed in actual table is it reflected in view?** Yes  **5.If data is changed in View is it reflected in actual table?** No   **Index** **1.What are Indexes?** Indexes is similar to indexes in dictionary.it helps in finding the data quickly.Therefore indexes are created on columns which are accessed frequently, so that the information can be retrieved quickly. Indexes can be created on a single column or a group of columns. When a index is created, it first sorts the data and then it assigns a ROWID for each row.  **2. How do you create Indexes?** Create index A on Student (Name, Age)   **3.What are diff types of index in SQL?** Clustered index- Reorders the records in the way they are created in the table. The logical grouping of table is similar to physical grouping. A table can have only one clustered index.  Non clustered index-he logical order of the index does not match the physical stored order of the rows on disk. A table can have more than one clustered index.  Note--i personally found this website really good for sql -http://www.w3schools.com |