1. Which are the components Of framework?

Common Language Runtime (CLR): The runtime environment that manages the execution of .NET code and provides services such as memory management and exception handling.

Base Class Library (BCL): A large collection of pre-written classes that provide functionality for tasks such as data access, security, and file manipulation.

Windows Forms: A library for creating graphical user interface (GUI) applications for Windows.

ASP.NET: A framework for building web applications, which includes libraries for creating web pages, handling HTTP requests, and managing session state.

ADO.NET: A data access technology that allows interaction with different databases.

Windows Communication Foundation (WCF): A framework for building service-oriented applications that can communicate over a variety of protocols.

Windows Workflow Foundation (WWF): A framework for building workflow-enabled applications.

LINQ: A technology for querying and manipulating data in a variety of formats, including SQL databases, XML documents, and in-memory collections.

XAML: A markup language used for creating user interfaces, which can be used with Windows Presentation Foundation (WPF) and Universal Windows Platform (UWP) applications.

2. What is assembly? What is manifest?

An assembly is a collection of one or more executable files (EXEs or DLLs) that are packaged together and can be deployed as a single unit. An assembly contains not only the code of an application or library, but also metadata that describes the assembly's version, culture, and security requirements.

A manifest is a part of an assembly that contains metadata about the assembly itself. The manifest includes information such as the assembly's name, version number, and list of dependencies on other assemblies. It also includes information about the types and resources that the assembly contains, and the security permissions it requires.

The manifest is stored in a portable executable (PE) file along with the assembly's code, and is used by the common language runtime (CLR) to load and execute the assembly. The manifest is an important part of the .NET assembly and plays an important role in the overall security and deployment of .NET assemblies.

3.What is GAC? How it handles DLL hell problem?

GAC stands for Global Assembly Cache. It is a location in the Windows operating system where shared assemblies are stored. The GAC is used to store assemblies that are shared by multiple applications on a machine. When a .NET application references an assembly, the runtime first checks the GAC to see if the assembly is already present there. If the assembly is found in the GAC, the runtime uses that version, otherwise it uses the version that is specified in the application's configuration file.

The GAC helps to resolve the "DLL hell" problem, which is a common issue in traditional Windows development. DLL hell occurs when multiple versions of a DLL are installed on a machine, and different applications use different versions of the DLL, leading to conflicts and errors.

4.What is the difference between sealed override and abstract

"sealed" is used to prevent a class or method from being inherited or overridden. When a class is declared as "sealed", it cannot be used as a base class for any other class. When a method is declared as "sealed", it cannot be overridden in any derived class.

"override" is used to indicate that a method in a derived class is intended to override a method with the same name and signature in the base class. When a method is declared as "override", the compiler checks that the method being overridden is marked as "virtual" or "abstract" in the base class.

"abstract" is used to define a class or method that has no implementation and must be implemented by a derived class. An abstract class can not be instantiated and it can contain abstract methods. An abstract method is a method with no implementation, it only contains method signature.

5.What is delegate? Explain the types of delegate?

A delegate is a type that represents a method with a specific signature. A delegate is similar to a function pointer in C or C++, but is safer and more type-safe. Delegates are used to pass methods as arguments to other methods, and can be used to define callback methods and event handlers.

There are two types of delegates in C#:

Single-cast delegate: A single-cast delegate can hold a reference to a single method. When invoked, the delegate calls the method it references.

Multi-cast delegate: A multi-cast delegate can hold references to multiple methods, and when invoked, calls all the methods it references.

Single-cast delegate is also known as "Function pointer" and Multi-cast delegate is also known as "Event"

6.What the difference is between finalize and dispose? Where using keyword is used?

In C#, both "finalize" and "Dispose" are used to release resources that are no longer needed by an object, but they are used in different situations and have different characteristics:

"finalize" is a method that is called by the garbage collector when an object is being collected. The garbage collector automatically calls the finalize method when it determines that an object is no longer reachable. The finalize method is used to release unmanaged resources that are associated with the object, such as file handles or memory allocated by native code.

"Dispose" is a method that is intended to be called by the user of an object when the object is no longer needed. The Dispose method is used to release both managed and unmanaged resources that are associated with the object. Unlike the finalize method, the Dispose method is not guaranteed to be called by the garbage collector.

The "using" keyword is used to create a block of code in which an object is instantiated and then automatically disposed of when the block of code is exited. The "using" statement is used to ensure that the Dispose method is called on an object, even if an exception is thrown.

8.Explain ASP .NET architecture with IIS7.

ASP.NET is a web application framework that runs on top of the .NET framework and is used for building dynamic web sites, web applications, and web services. The architecture of ASP.NET is based on a three-tier model, which consists of:

Presentation Layer: This is the top-most layer of the architecture and is responsible for displaying the user interface to the user. It includes web pages, user controls, and web forms.

Business Logic Layer: This layer contains the business logic and is responsible for processing the data that is sent from the presentation layer. It includes classes, components, and services that implement the business logic of the application.

Data Access Layer: This layer is responsible for interacting with the database and performing tasks such as inserting, updating, and retrieving data. It includes classes, components, and services that interact with the database.

ASP.NET is a web application framework that runs on top of the .NET framework and is used for building dynamic web sites, web applications, and web services. The architecture of ASP.NET is based on a three-tier model, which consists of:

Presentation Layer: This is the top-most layer of the architecture and is responsible for displaying the user interface to the user. It includes web pages, user controls, and web forms.

Business Logic Layer: This layer contains the business logic and is responsible for processing the data that is sent from the presentation layer. It includes classes, components, and services that implement the business logic of the application.

Data Access Layer: This layer is responsible for interacting with the database and performing tasks such as inserting, updating, and retrieving data. It includes classes, components, and services that interact with the database.

ASP.NET applications run on top of the Internet Information Services (IIS) web server, which is a component of the Windows operating system. IIS 7 is the latest version of IIS and includes several new features and improvements over previous versions, including:

Improved support for web standards Increased security Improved management Integrated pipeline

9.Explain steps of ADO .NET program.

ADO.NET is a data access technology that allows you to interact with databases from your .NET application. The steps of an ADO.NET program typically involve the following:

Connect to the database MYsql or MSSql

Execute a command : The command can be a SQL statement or a stored procedure, and can be used to perform tasks such as inserting, updating, and retrieving data.

Retrieve data: Use a data reader to retrieve data from the database. Data readers provide a forward-only, read-only view of the data, and are typically used for retrieving large amounts of data.

Work with data: Use a DataSet or DataTable to work with the data retrieved from the database. A DataSet can contain multiple DataTable objects, and can be used to work with disconnected data. DataTable is used to work with tabular data.

Close the connection: Close the connection to the database when you are finished with it. This releases the resources that were used to connect to the database, and ensures that the connection is available for other operations.

Optionally, use a DataAdapter to fill the DataSet or DataTable with the data retrieved from the database and also use it to update the changes made in DataSet or DataTable to the Database.

10.Explain MVC application life cycle?

The Model-View-Controller (MVC) application life cycle is the process that an MVC web application goes through from the time a user requests a page to the time the response is sent back to the user. The MVC life cycle can be broken down into the following steps:

Routing: When a user requests a page, the routing system examines the URL of the request and maps it to a specific controller and action method.

Instantiation of controller: The MVC framework creates an instance of the specified controller class.

Execution of action method: The specified action method on the controller is executed. This method is responsible for handling the user's request and preparing the data that will be displayed in the view.

Execution of action filter: If there are any action filters on the action method, they are executed. These filters are used to perform additional actions before or after the execution of an action method.

Instantiation of view: The MVC framework creates an instance of the specified view.

Execution of view: The specified view is executed and the final HTML is rendered to the user.

Execution of result filter: If there are any result filters on the action method, they are executed. These filters are used to perform additional actions before or after the execution of a view.

Sending of response: The HTML generated by the view is sent back to the user as the response.

11.Explain what is routing in MVC? What are the three segments for routing important?What is default route?

In MVC (Model-View-Controller), Routing is the process of mapping URLs to specific controllers and actions. Routing is used to determine which controller and action method should handle a request based on the URL of the request.

Routes are defined in the routing table, which is a collection of routes that are used to map URLs to controllers and actions. Each route in the routing table consists of three segments:

The URL pattern: This is a string that defines the pattern of the URL that the route should match. The URL pattern can include placeholders for variables, such as {controller} and {action}, which are used to specify the controller and action that should handle the request.

The default values: These are the default values that should be used for the variables in the URL pattern if they are not specified in the URL. For example, the default value for the {controller} variable might be "Home" and the default value for the {action} variable might be "Index".

Constraints: These are used to constrain the pattern matching of the route. These are regular expressions or custom functions that are used to validate the values of the variables in the URL.

12. What are Filters in MVC? Explain action filters.

In MVC, filters are used to add extra functionality to controllers and actions. They allow developers to execute code before or after a specific action is executed, or even to prevent an action from executing altogether. There are several types of filters in MVC, including action filters, authentication filters, authorization filters, exception filters, and result filters.

An action filter is a filter that is executed before or after an action is executed. They can be used to perform tasks such as logging, caching, or modifying the action's parameters. Action filters are implemented as attributes that can be applied to controllers or actions.

There are several built-in action filters in MVC, including:

OutputCache: caches the output of an action for a specified amount of time HandleError: handles exceptions that occur during the execution of an action and returns a predefined error view Authorize: checks that the current user is authorized to access the action ValidateAntiForgeryToken: validates that a request is not a forgery by checking a token that is sent with the request.

13.What are the methods Of handling an Error in MVC?

In MVC (Model-View-Controller) there are several ways to handle errors:

Using try-catch blocks: Errors can be handled in the controller's action methods or in the business logic layer using try-catch blocks. When an exception is caught, the code in the catch block can be used to handle the error and redirect the user to an error page or display an error message.

Using HandleErrorAttribute: The HandleErrorAttribute is an MVC filter that can be applied to a controller or action method. It is used to handle errors that occur during the execution of the controller or action method. When an error occurs, the HandleErrorAttribute will redirect the user to an error view.

Using Application_Error event in global.asax: The Application_Error event is raised when an unhandled exception occurs in an MVC application. The event handler can be used to handle the error and redirect the user to an error page or display an error message.

Using customErrors in web.config: The customErrors element in the web.config file can be used to configure how errors are handled in an MVC application.

14.Differences between Razor and ASPX View Engine in MVC?

Razor and ASPX are both view engines in MVC (Model-View-Controller) that are used to generate the HTML that is sent to the browser. They are both used to parse the views and combine them with the data from the model to generate the final HTML. However, they have some differences:

Syntax: The most notable difference between Razor and ASPX is the syntax. Razor uses a more compact and expressive syntax, which is designed to be easy to read and write. It uses the "@" symbol to denote code blocks and uses C# or Visual Basic as its programming language. On the other hand, ASPX uses a more traditional and verbose syntax, which is similar to traditional ASP. It uses the "<% %>" tags to denote code blocks and uses C# or Visual Basic as its programming language.

Intellisense: Razor views have better Intellisense support compared to ASPX views. Razor views support both C# and Visual Basic and provide better Intellisense for both languages. On the other hand, ASPX views only support C# or Visual Basic, and the Intellisense support is not as good as Razor views.

Simplicity: Razor views are simpler and more lightweight compared to ASPX views. Razor views are designed to be easy to read and write, and are optimized for server-side code. On the other hand, ASPX views are more complex and verbose, and are optimized for client-side code.

Performance: Razor views generally have better performance compared to ASPX views. Razor views are designed to be more lightweight and optimized for server-side code, and they have better performance when it comes to rendering views.

15.Explain .NET core features and architecture.

.NET Core is a cross-platform, open-source, and modular version of the .NET framework. It was designed to provide a more lightweight, flexible, and performant option for building modern applications. Some of the key features of .NET Core include:

Cross-platform compatibility: .NET Core runs on Windows, Linux, and macOS, which allows developers to write code once and run it on multiple platforms.

Modularity: .NET Core is designed as a set of NuGet packages, which allows developers to only include the packages they need in their application. This reduces the application's size and improves performance.

Flexibility: .NET Core allows developers to create and run applications in a variety of different environments, including the command line, web, and cloud.

Performance: .NET Core has been optimized for performance, and it includes features such as Just-In-Time (JIT) compilation, Ahead-of-Time (AOT) compilation and runtime optimization that improves the runtime performance of an application.

Open-source: .NET Core is an open-source platform, which means that the source code is available for anyone to use, modify, and distribute.