

Interview questions

.NET FRAMEWORK
ASP.NET MVC

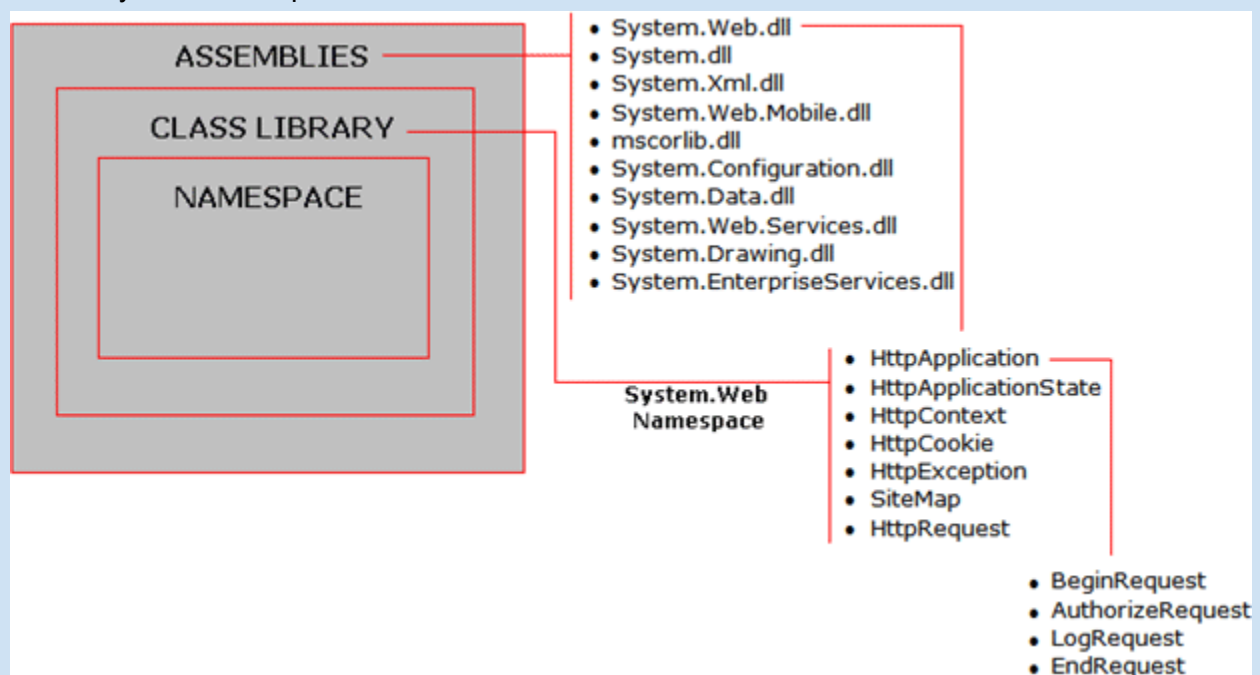
Basic .NET Framework

Name present version of .net ?

NET Framework 4.8. 1 is included in the latest version of Visual Studio, Visual Studio 2022

17.3. .09-Aug-2022

Assembly and Namespace difference?



GAC please explain in detail?

Advantages of .net ?

Reflections explain their uses?

Assemblies where can we view them(folder)?

JIT how many types are there .Explain?

Url -how many parts of urls are there?

(B)What is an IL?.....	28
(B)What is a CLR?.....	28
(B)What is CTS?.....	29
(B)What is a CLS (Common Language Specification)?.....	29
(B)What is a Managed Code?.....	29
(B)What is a Assembly?	29

(A) What are the different types of Assembly?	30
(B) What is NameSpace?.....	30
(A) What is Manifest?.....	32
(B) Where is version information stored of an assembly?.....	32
(I) Is versioning applicable to private assemblies?.....	32
(B) What is GAC?.....	32
(I) what is the concept of strong names?	32
(I) How to add and remove an assembly from GAC?	35
(B) What is Delay signing?.....	35
(B) What is garbage collection?.....	36
(I) Can we force garbage collector to run?	36
(B) What are Value types and Reference types?	38
(B) What is concept of Boxing and Unboxing ?.....	38
(B) What is the difference between VB.NET and C#?.....	38
(I) what is the difference between System exceptions and Application exceptions?	39
(I)What is CODE Access security?.....	40
(I)What is a satellite assembly?	40
(A) How to prevent my .NET DLL to be decompiled?	40
(I) what is the difference between Convert.ToString and .toString () method?.....	41
(A) What is Native Image Generator (Ngen.exe)?	41
(A) If we have two version of same assembly in GAC how do we make a choice?	42
(A)What is CodeDom?	48
Chapter 2: NET Interoperability.....	50
(I) How can we use COM Components in .NET?	50
(I) We have developed the COM wrapper do we have to still register the COM?... ..	51
(A)How can we use .NET components in COM?.....	51
(A) How can we make Windows API calls in .NET?.....	54
(B) When we use windows API in .NET is it managed or unmanaged code?	56
(I)What is COM?	56
(A) What is Reference counting in COM?	56
(A) Can you describe IUNKNOWN interface in short?.....	56
(I) Can you explain what DCOM is?	56
(B) How do we create DCOM object in VB6?.....	57
(A) How to implement DTC in .NET?	57
(A) How many types of Transactions are there in COM + .NET?	59
(A) How do you do object pooling in .NET?.....	59
(A) What are types of compatibility in VB6?.....	60
(A)What is equivalent for regsvr32 exe in .NET?	61

Chapter 3: Threading	61
(B) What is Multi-tasking?	61
(B) What is Multi-threading?.....	61
(B) What is a Thread?.....	61
(B) Did VB6 support multi-threading?.....	61
(B)Can we have multiple threads in one App domain?	61
(B) Which namespace has threading?.....	62
(A) What does Address Of operator do in background?.....	63
(A) How can you reference current thread of the method?	63
(I) what is Thread.Sleep () in threading?	63
(A) How can we make a thread sleep for infinite period?	63
(A) What is Suspend and Resume in Threading?	63
(A) What the way to stop a long running thread?.....	64
(A) How do I debug thread?	64
(A) What is Thread.Join () in threading?.....	64
(A) What are Daemon threads and how can a thread be created as Daemon?	65
(A) How is shared data managed in threading?.....	65
(I) Can we use events with threading?.....	65
(A) How can we know a state of a thread?	65
(A) What is use of Interlocked class ?	65
(A) What is a monitor object?.....	65
(A) What are wait handles?.....	66
(A) What is ManualResetEvent and AutoResetEvent?	66
(A) What is Reader Writer Locks?	66
(I) How can you avoid deadlock in threading?.....	66
(B) What is the difference between thread and process?.....	67
Chapter 4: Remoting and Webservices.....	67
(B)What is an application domain?.....	67
(B) What is .NET Remoting?.....	67
(B) Which class does the remote object has to inherit?.....	68
(I) what are two different types of remote object creation mode in .NET ?.....	68
(A) Describe in detail Basic of SAO architecture of Remoting?.....	68
(A) What are the situations you will use singleton architecture in remoting?.....	73
(A) What is fundamental of published or precreated objects in Remoting?.....	73
(A) What are the ways in which client can create object on server in CAO model?	73
(A) Are CAO stateful in nature?.....	73
(A) To create objects in CAO with 'new' keyword what should be done?.....	74
(I) Is it a good design practice to distribute the implementation to Remoting Client?	74
(A) What are LeaseTime, SponsorshipTime, RenewonCallTime and LeaseManagerPollTime?	75
(A) Which config file has all the supported channels/protocol?.....	75
(A) How can you specify remoting parameters using Config files?.....	76
(A) Can Non-Default constructors be used with Single Call SAO?.....	78

(I) How can we call methods in remoting asynchronously?.....	79
(A) What is Asynchronous One-Way Calls?.....	79
(B) What is marshalling and what are different kinds of marshalling?	79
(A) What is ObjRef object in remoting?.....	79
(B) What is a Web Service?.....	80
(B) What is UDDI?	80
(B) What is DISCO?	80
(B) What is WSDL?.....	80
(A) What the different phase/steps of acquiring a proxy object in Web service?	80
(A) What the different phase/steps of acquiring a proxy object in Web service?	81
(B) What is file extension of Web services?.....	81
(B) Which attribute is used in order that the method can be used as WebService? .	82
(A) What are the steps to create a web service and consume it?	82
(A) Do webservice have state?	87
Chapter 5: Caching Concepts	88
(B) What is an application object?.....	88
(I) what is the difference between Cache object and application object?.....	88
(I) How can get access to cache object?	88
(A) What are dependencies in cache and types of dependencies?.....	88
(A) Can you show a simple code showing file dependency in cache?	88
(A) What is Cache Callback in Cache?.....	89
(A) What is scavenging?.....	89
(B) What are different types of caching using cache object of ASP.NET?.....	90
(B) How can you cache different version of same page using ASP.NET cache object?.....	90
(A) How will implement Page Fragment Caching?.....	90
(B) Can you compare ASP.NET sessions with classic ASP?.....	90
(B) Which are the various modes of storing ASP.NET session?.....	91
(A) Is Session_End event supported in all session modes?	91
(A) What are the steps to configure StateServer Mode?.....	91
(A) What are the steps to configure SQLServer mode?	91
(A) Where do you specify session state mode in ASP.NET?.....	92
(B) What are the other ways you can maintain state?.....	92
(B) What are benefits and Limitation of using Hidden fields?.....	92
(B) What is ViewState?	92
(A) Does the performance for viewstate vary according to User controls?.....	93
(B) What are benefits and Limitation of using Viewstate for state management? ...	93
(B) How can you use Hidden frames to cache client data ?	93
(I) What are benefits and limitations of using Hidden frames?.....	94
(I) What are benefits and limitations of using Cookies?.....	94
(I) What is Query String and What are benefits and limitations of using Query Strings?	94
(I) What is Absolute and Sliding expiration?	95
(I) What is cross page posting?.....	95

(I) How do we access viewstate value of this page in the next page ?	96
(I) Can we post and access view state in another application?.....	96
(I) What is SQL Cache Dependency in ASP.NET 2.0?.....	96
(I) How do we enable SQL Cache Dependency in ASP.NET 2.0?	96
(I) What is Post Cache substitution?.....	102
(I) Why do we need methods to be static for Post Cache substitution?.....	104
Chapter 6: OOPS	104
What is the difference between Console.Read(),Console.ReadLine() and Console.ReadKey()?	

The program to differentiate Read vs Readline vs Readkey in C#.

The Read method : Reads the **next character** from the standard input stream .

ReadLine method : Reads the **next line of characters** from the standard input stream.

ReadKey method : Obtains the **next character or function** key pressed by the user.The **pressed key is displayed** in the console window. Let's See Program.

```
using System;

namespace ConsoleApp9
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter Any String : ");
            string str1 = Console.ReadLine();
            Console.WriteLine("Entered String " + str1 + "\n");

            Console.WriteLine("Enter Any Integer :");
            int str = Console.Read();
            Console.Write(str.GetType()); // Return DataType

            Console.WriteLine("\n"+ "Enter Any Key ");
            ConsoleKeyInfo k = Console.ReadKey();
            Console.WriteLine("\n");
            Console.Write("Key Info {0} ", k.Key );
        }
    }
}
```

```
using System;

namespace ConsoleApp9
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Enter Any String : ");
            string str1 = Console.ReadLine();
            Console.WriteLine("Entered String " + str1 + "\n");

            Console.WriteLine("Enter Any Integer :");
            int str = Console.Read();
            Console.WriteLine(str.GetType());

            Console.WriteLine("\n" + "Enter Any Key ");
            ConsoleKeyInfo k = Console.ReadKey();
            Console.WriteLine("\n");
            Console.WriteLine("Key Info {0} ", k.Key );
        }
    }
}
```

Microsoft Visual Studio Debug Console

Enter Any String :
Hello
Entered String Hello

Enter Any Integer :
9
System.Int32

Enter Any Key
m
Key Info M

(B) What is Object Oriented Programming?	104
(B) What is a Class?.....	104
(B) What is an Object?.....	104
(A) What is the relation between Classes and Objects?	104
What are access modifiers in C#?	

Caller's location	public	protected internal	protected	internal	private protected	private
Within the class	✓	✓	✓	✓	✓	✓
Derived class (same assembly)	✓	✓	✓	✓	✓	✗
Non-derived class (same assembly)	✓	✓	✗	✓	✗	✗
Derived class (different assembly)	✓	✓	✓	✗	✗	✗
Non-derived class (different assembly)	✓	✗	✗	✗	✗	✗

(B) What are different properties provided by Object-oriented systems?	104
(B) How can we achieve inheritance in VB.NET?.....	105
(I) what are abstract classes?.....	107
(B) What is a Interface?	109
(A) What is difference between abstract classes and interfaces?	110
(B) What is a delegate?.....	110
(B) What are Events?.....	111
(I) Do events have return type.	113
(A) Can events have access modifiers?	113
(A) Can we have shared events?.....	113
(I) what is shadowing?.....	113
(A) What is the difference between Shadowing and Overriding?	114
(I) what is the difference between delegate and events?.....	114
(B) If we inherit a class do the private variables also get inherited?	114
(B) What is the different accessibility levels defined in .NET?	114
(I) Can you prevent a class from overriding?	115
(I) what is the use of "Must inherit" keyword in VB.NET?	115
(I) Do interface have accessibility modifier.....	115
(A) What are similarities between Class and structure?	115
(A) What is the difference between Class and structure's?	115
(B) What does virtual keyword mean?	115

(B) What are shared (VB.NET)/Static(C#) variables?	115
(B) What is Dispose method in .NET?	117
(B) What is the use of “Overrides” and “Overridable” keywords?	117
(A)Where are all .NET Collection classes located?.....	117
(A) What is ArrayList?	117
(A) What is a HashTable?.....	117
(A) What are queues and stacks?.....	117
(B) What is ENUM?	118
(A) What is nested Classes?.....	118
(B)What is Operator overloading in .NET?.....	118
(I) For the below code which constructor will fire first?.....	118
(B)What is the significance of Finalize method in .NET?.....	119
(I)How can we suppress a finalize method?	120
(B)What is the use of DISPOSE method?	120
A) How do I force the Dispose method to be called automatically, as clients can forget to call Dispose method?	120
(I) In what instances you will declare a constructor to be private?	121
(I) Can we have different access modifiers on get/set methods of a property ?	121
(I) If we write a goto or a return statement in try and catch block will the finally block execute?.....	121
(A) What is Indexer?.....	121
(A) Can we have static indexer in C#?	121
(A)Can two catch blocks be executed?.....	121
(A) What is the difference between System.String and System.StringBuilder classes?.....	121
Chapter 7: ASP.NET.....	121
(B) What' is the sequence in which ASP.NET events are processed?	121
(B) In which event are the controls fully loaded?.....	122
(B) How can we identify that the Page is Post Back?.....	122
(B) How does ASP.NET maintain state in between subsequent request?	122
(A) What is event bubbling?.....	122
B) How do we assign page specific attributes?	122
(A) How do we ensure viewstate is not tampered?.....	122
(B) What is the use of @ Register directives?.....	122
(B) What is the use of Smart Navigation property?.....	122
(B) What is AppSetting Section in “Web.Config” file?	123
(B) Where is View State information stored?.....	123
(I) what is the use of @ Output Cache directive in ASP.NET.....	123
(B) How can we create custom controls in ASP.NET?	123
(B) How many types of validation controls are provided by ASP.NET?.....	123
(B) Can you explain “AutoPostBack”?.....	124
(B) How can you enable automatic paging in Data Grid?.....	124
(B) What is the use of “GLOBAL.ASAX” file?.....	124
(B) What is the difference between “Web.config” and “Machine.Config”?.....	124

(B) What is a SESSION and APPLICATION object?	124
(A) What is the difference between 'Server.Transfer' and 'response.Redirect' ?.	124
(A)What is the difference between Authentication and authorization?.....	125
(I) what is impersonation in ASP.NET?	125
(B) Can you explain in brief how the ASP.NET authentication process works? ...	125
(A) What are the various ways of authentication techniques in ASP.NET?	126
(A)How does authorization work in ASP.NET?.....	127
(B)What is difference between Data grid, Datalist, and repeater?	128
(A)From performance point of view, how do they rate?	128
(B)What is the method to customize columns in Data Grid?	128
(B)How can we format data inside Data Grid?.....	128
(A) How to decide on the design consideration to take a Data grid, data list, or repeater?.....	128
(B) Difference between ASP and ASP.NET?.....	129
(A) What are major events in GLOBAL.ASAX file?.....	130
(A) What order they are triggered?.....	131
(I) Do session use cookies?.....	131
(I)How can we force all the validation control to run?.....	131
(B)How can we check if all the validation control are valid and proper?	131
(A) If client side validation is enabled in your Web page, does that mean server side code is not run.....	132
(A)Which JavaScript file is referenced for validating the validators at the client side?	132
(B)How to disable client side script in validators?	132
(A)How can I show the entire validation error message in a message box on the client side?	132
(B)You find that one of your validations is very complicated and does not fit in any of the validators, what will you do?.....	132
(I)What exactly happens when ASPX page is requested from a browser?.....	133
(B) How can we kill a user session?	136
(I) How do you upload a file in ASP.NET?.....	136
(I) How do I send email message from ASP.NET?.....	136
(A)What are different IIS isolation levels?.....	136
(A)ASP used STA threading model, what is the threading model used for ASP.NET.	137
(A)What is the use of <%@ page aspcompat=true %> attribute?.....	137
B) Explain the differences between Server-side and Client-side code?	138
(I)Can you explain Forms authentication in detail?.....	138
(A)How do I sign out in forms authentication?	139
(A)If cookies are not enabled at browser end does form Authentication work?	139
(A)How to use a checkbox in a data grid?.....	139
(I)What are the steps to create a windows service in VB.NET?.....	140
(A) What is the difference between "Web farms" and "Web garden"?	141
(A) How do we configure "Web Garden"?.....	143

(B) What is the main difference between Grid layout and Flow Layout?.....	145
(I) What's the difference between trace and debug in ASP.NET?	145
(A) How do you enable tracing in on an ASP.NET page?	146
(B) Which namespace is needed to implement debug and trace ?.....	147
(A) Can you explain the concept of trace listener?.....	147
(I) What are trace switches?.....	148
Chapter 8: NET Architecture.....	148
(B) What are design patterns?.....	149
(A) What is the difference between Factory and Abstract Factory Patterns?.....	150
(I)What is MVC pattern?	152
(A)How can we implement singleton pattern in .NET?	153
(A)How do you implement prototype pattern in .NET?.....	154
(I)What are the situations you will use a Web Service and Remoting in projects?	154
(A)Can you give a practical implementation of FAÇADE patterns?	154
(I) How can we implement observer pattern in .NET?.....	155
(B)What is three-tier architecture?.....	155
(I)Have you ever worked with Microsoft Application Blocks, if yes then which?	156
(A)What is Service Oriented architecture?.....	156
(I)What are different ways you can pass data between tiers?.....	158
(A)What is Windows DNA architecture?.....	158
(A)What is aspect oriented programming?	159
Chapter 9: ADO.NET	166
(B) What is the namespace in which .NET has the data functionality class?.....	166
(B) Can you give an overview of ADO.NET architecture?.....	166
(B) What are the two fundamental objects in ADO.NET?	167
(B) What is difference between dataset and data reader?	167
(I) What are major difference between classic ADO and ADO.NET?.....	167
(B) What is the use of connection object?	168
(B) What is the use of command objects?	168
(B)What is the use of data adapter?	168
(B)What are basic methods of Data adapter?	168
(B) What is Dataset object?	169
(B) What are the various objects in Dataset?.....	169
(B) How can we connect to Microsoft Access, FoxPro, and Oracle etc?.....	169
(B) How do we connect to SQL SERVER, which namespace do we use?	170
(B) How do we use stored procedure in ADO.NET and how do we provide parameters to the stored procedures?.....	174
(B) How can we force the connection object to close after my data reader is closed?	175
(B) I want to force the data reader to return only schema of the data store rather than data.....	175
(B) How can we fine-tune the command object when we are expecting a single row?	175
(B) Which is the best place to store connection string in .NET projects?	176

(B) What are the steps involved to fill a dataset?	176
(B)What are the various methods provided by the dataset object to generate XML?	177
(B) How can we save all data from dataset?.....	177
(B) How can we check that some changes have been made to dataset since it was loaded?	177
(B) How can we add/remove row is in “Data Table” object of “Dataset”?.....	178
(B) What is basic use of “Data View”?	178
(B) What is the difference between “Dataset” and “Data Reader” ?.....	178
(B) How can we load multiple tables in a Dataset?.....	179
(B) How can we add relation between tables in a Dataset?.....	179
(B) What is the use of Command Builder?.....	180
(B) What’s difference between “Optimistic” and “Pessimistic” locking ?.....	180
(A) How many ways are there to implement locking in ADO.NET?.....	180
(A)How can we perform transactions in .NET?	180
(I) What is difference between Dataset? Clone and Dataset. Copy?	181
(A) Can you explain the difference between an ADO.NET Dataset and an ADO Record set?.....	181
(A) Explain in detail the fundamental of connection pooling?.....	181
(A)What is Maximum Pool Size in ADO.NET Connection String?	182
(A)How to enable and disable connection pooling?.....	182
(I) What extra features does ADO.Net 2.0 have ?	183
Chapter 10: SQL SERVER.....	183
(B) What is normalization? What are different types of normalization?.....	183
(B) What is denormalization?	185
(B) What is a candidate key?	185
(B) What are the different types of joins? What is the difference between them? .	186
(I)What are indexes? What is the difference between clustered and nonclustered indexes?.....	186
(A)How can you increase SQL performance?.....	186
(A)What is the use of OLAP?.....	187
(A)What is a measure in OLAP?	187
(A)What are dimensions in OLAP?.....	187
(A)What are levels in dimensions?.....	187
(A)What are fact tables and dimension tables in OLAP?	188
(A)What is DTS?	188
(A)What is fill factor ?.....	188
(A)What is RAID and how does it work?.....	188
(B)What is the difference between DELETE TABLE and TRUNCATE TABLE commands?	189
(B)If locking is not implemented, what issues can occur?	189
(B)What are different transaction levels in SQL SERVER?	190
(I)What are the different locks in SQL SERVER?	191
(I) Can we suggest locking hints to SQL SERVER?.....	191

(I) What is LOCK escalation?.....	192
(B) What are the different ways of moving data between databases in SQL Server?	192
(I) What are advantages of SQL 2000 over SQL 7.0?	192
(B) What is the difference between a HAVING CLAUSE and a WHERE CLAUSE?	193
(B) What is the difference between UNION and UNION ALL SQL syntax?	193
(I) How can you raise custom errors from stored procedure?	193
(I) what is ACID fundamental? What are transactions in SQL SERVER?	194
(A) What is DBCC?.....	195
(A) What is the purpose of Replication?.....	196
(A) What are the different types of replication supported by SQL SERVER?	196
(I) What is BCP utility in SQL SERVER?	197
(I)What are the different types of triggers in SQL SERVER?.....	198
(A)If we have multiple AFTER Triggers on table how can we define the sequence of the triggers?.....	198
(A)What is SQL injection?	198
(B) What is the difference between Stored Procedure (SP) and User Defined Function (UDF)?.....	199
Chapter 11: UML.....	199
(B) What is UML?	199
(I) How many types of diagrams are there in UML?.....	199
(B) What are advantages of using UML?	201
(A)What is the sequence of UML diagrams in project?	201
(A)Give a small brief explanation of all Elements in activity diagrams?.....	204
(A)Explain Different elements of a collaboration diagram?.....	206
(A) Explain all parts of a deployment diagram?	208
(A) Describe the various components in sequence diagrams?.....	209
(A) What are the elements in State Chart diagrams?	210
(A)Describe different elements in Static Chart diagrams?	211
(A)Explain the different elements of a Use Case?.....	213
Chapter 12: Project Management	214
(B) What is project management?.....	214
(A) Is spending in IT projects constant through out the project?	214
(B) Who is a stakeholder?.....	215
(B) Can you explain project life cycle?	215
(B) Are risk constant through out the project?.....	216
(A) Can you explain different software development life cycles?	217
(B) What is triple constraint triangle in project management?.....	219
(B) What is a project baseline?	219
(B) What is effort variance?.....	219
(B) How is normally a project management plan document organized?.....	219
(I)How do you estimate a project?.....	220
(B)What is CAR (Causal Analysis and Resolution)?	220

(B) What is DAR (Decision Analysis and Resolution)?.....	220
(B) What is a fish bone diagram?	220
(B) What is Pareto principle?.....	221
(B) How do you handle change request?	221
(I) What is internal change request?	222
(B) What is difference between SITP and UTP in testing?	222
(B) Which software have you used for project management?.....	222
(I) What are the metrics followed in project management?.....	222
(B)People in your project do not perform , what will you do?	224
(B)What is black box testing and White box testing?.....	224
(B) What is the difference between Unit testing, Assembly testing and Regression testing?	225
(I) What is V model in testing?.....	227
(B)How do you start a project?.....	228
(B)How did you do resource allocations?.....	228
(I) How will you do code reviews?.....	228
(A)What is CMMI?.....	229
(A) What are the five levels in CMMI?.....	229
(A) What is continuous and staged representation?.....	231
(A)What is SIX sigma?.....	247
(A)What are DMAIC and DMADV?	248
(A)What are the various roles in Six Sigma implementation?	248
(I)What are function points?	249
(I)What are the different types of elementary process in FPA?.....	249
(I)What are the different elements in Functions points?.....	250
(A) Can you explain in GSC and VAF in function points?	254
(I) What are unadjusted function points and how is it calculated?.....	255
(I) Can you explain steps in function points?	256
(I) What is the FP per day in your current company?.....	257
(A)Do you know Use Case points?.....	257
(A)What is COCOMO I, COCOMOII and COCOMOIII?	257
(A) What is SMC approach of estimation?.....	257
(A)How do you estimate maintenance project and change requests?.....	257
Chapter 13: XML.....	257
(B) What is XML?	257
(I) What is the version information in XML?.....	258
(B) What is ROOT element in XML?.....	258
(B) If XML does not have closing tag will it work?.....	258
(B) Is XML case sensitive?.....	258
(B) What is the difference between XML and HTML?.....	258
(B) Is XML meant to replace HTML?.....	258
(A) Can you explain why your project needed XML?	258
(B) What is DTD (Document Type Definition)?.....	258
(B) What is well formed XML?.....	258

(B) What is a valid XML?	258
(B) What is CDATA section in XML?.....	259
(B) What is CSS?.....	259
(B) What is XSL?.....	259
(B) What is element and attributes in XML?.....	259
(B) Which are the namespaces in .NET used for XML?	259
(A) What are the standard ways of parsing XML document?	259
(A)In What scenarios will you use a DOM parser and SAX parser?	261
(A) How was XML handled during COM times?	262
(A)What is the main difference between MSML and .NET Framework XML classes?.....	262
(B) What are the core functionalities in XML .NET framework? Can you explain in detail those functionalities?.....	262
(B)What is XSLT?	262
(I) Define XPATH?.....	263
(A) What is the concept of XPOINTER?	263
(B) What is an XMLReader Class?	263
(B) What is XMLTextReader?.....	263
(I) How do we access attributes using “XmlReader”?.....	264
(I) Explain simple Walk through of XmlReader.....	264
(A) What does XmlValidatingReader class do?	266
Chapter 14: Localization/Globalization.....	267
(B) What is Unicode & Why was it introduced?	267
(I) Does .NET support UNICODE and how do you know it supports?	268
(A) What is the difference between localization and globalization?	268
(A)What architecture decisions you should consider while planning for international software's?	268
(I) How do we get the current culture of the environment in windows and ASP.NET?.....	270
(B) Which are the important namespaces during localization and globalization?..	273
(B) What are resource files and how do we generate resource files?	273
(I) Can resource file be in any other format other than resx extensions?	275
(I) How is resource files actually used in project?.....	275
(A) How can we use Culture Auto in project?.....	275
(B) What are satellite assemblies?	277
(A) How do we generate Satellite assemblies?.....	277
(A) What is AL.EXE and RESGEN.EXE?.....	277
(I) What is the use of resource manager class?.....	278
(A) What precautions do we need to take while deploying satellite assemblies? ..	279
(A) Can we get a strongly typed resource class rather than using resource manager?	279
(A) Can you explain the fundamentals of “GetGlobalResourceObject” and “GetLocalResourceObject” functions?.....	281
(A) Can we sign a satellite assembly?	282

(I) Can you explain collation sequence in sql server?	282
(A)How do we define collation sequence for database and tables?.....	282
(A)Can we change the order in a select query with a specified collation sequence?	283
(A) Can you list best practices for globalization and localization?	283
(A) Why is the culture set to the current thread?.....	284
Chapter 15: Windows Communication Foundation (Vista Series)	285
(I) What are the important principles of SOA (Service oriented Architecture)?....	286
(I) What are ends, contract, address, and bindings?	287
(A) Which specifications does WCF follow?	288
(A) What are the main components of WCF?	289
(I) Explain how Ends, Contract, Address, and Bindings are done in WCF?.....	289
(I) what is a service class?	289
(I) what is a service contract, operation contract and Data Contract?	289
(I) what are the various ways of hosting a WCF service?	294
(I) How do we host a WCF service in IIS?.....	295
(I) what are the advantages of hosting WCF Services in IIS as compared to selfhosting?.....	298
(I) what are the major differences between services and Web services?.....	299
(I) What is the difference WCF and Web services?	299
(A) What are different bindings supported by WCF?.....	299
(A) Which are the various programming approaches for WCF?.....	300
(A) What is one-way operation?.....	300
(A) Can you explain duplex contracts in WCF?.....	302
(A) How can we host a service on two different protocols on a single server?.....	304
(A) How can we use MSMQ bindings in WCF?	306
(A) Can you explain transactions in WCF?	309
(A) What different transaction isolation levels provided in WCF?	309
(A) Can we do transactions using MSMQ?	311
(A)Can we have two-way communications in MSMQ?.....	312
(A) What are Volatile queues?.....	312
(A) What are Dead letter queues?.....	313
(A) What is a poison message?.....	314
Chapter 16: Windows Presentation Framework (Vista Series)	314
(B) What is WPF?.....	314
(B) What is XAML?	314
(I) What are dependency properties?	315
(A) Are XAML file compiled or built on runtime?	315
(B) Can you explain how we can separate code and XAML?.....	315
B) How can we access XAML objects in behind code?.....	316
(A) What kind of documents are supported in WPF?.....	317
Chapter 17: Windows workflow foundation (Vista series)	317
(B) What is Windows Workflow Foundation?	317
(B) What is a Workflow?.....	317

(B) What are different types of Workflow in Windows Workflow foundation?....	318
(I) when should we use a sequential workflow and when should we use state machines?.....	319
(I) How do we create workflows using designer?	320
(I) How do we specify conditions in Work flow?.....	321
(I) How do you handle exceptions in workflow?.....	323
(I) What is the use of XOML files.....	324
(A) How can we pass parameters to workflow?	325
Chapter 18: ATLAS-AJAX	326
(B) What problem does Ajax solve?.....	326
(B) What is Ajax?	327
(B) What is the fundamental behind Ajax?.....	327
(B) What is JSON?.....	328
(B) How do we use XMLHttpRequest object in JavaScript?	328
(B) How do we do asynchronous processing using Ajax?	329
(B) What are the various states in XMLHttpRequest and how do we check the same?	329
(B) How can we get response text?.....	329
(B) How can we send request to the server using the XMLHttpRequest component?	329
(I) How do we pass parameters to the server?	330
(I) How can we create a class in JavaScript using Atlas?.....	330
(A) How do we do inheritance-using Atlas?.....	332
(A) How do we define interfaces using Atlas?	332
(A) How do we reference HTML controls using Atlas?.....	333
(I) Can you explain Scriptmanager control in Ajax?.....	334
(B) Can you explain Enablepartialrendering and UpdatePanel control in Ajax? ...	335
(I) Can you explain the concept of triggers in 'UpdatePanel' control?	337
(I) Can you explain the 'UpdateProgress' component?.....	338
(A) How can you do validations in Ajax?	339
(A) How do we do exception handling in Ajax?	339
(A) How do we consume web service in Atlas?.....	340
(A) How can we consume data directly in web services?	343
Chapter 19:- Reports.....	343
(B) How do we access crystal reports in .NET?	343
(I) What are the various components in crystal reports?	344
(I) What basic steps are needed to display a simple report in crystal?	345
(I) Can crystal reports be published as a web service?	348
(I) How do we invoke the crystal report web service?	349
(I) How do we add formulas using crystal reports?.....	349
(I) How do we pass parameters to crystal reports?.....	350
(I) How do we export from crystal reports?.....	351
(I) How do we print to printer using crystal?.....	352
(I) How do we generate cross tab reports?.....	352

(A) How can we do grouping in crystal?.....	353
(A) Can you explain three-pass reporting which crystal report uses?	353
(B) Can you explain reporting services architecture?.....	354
(B) We have two IIS application 'Reports' and 'Reportserver' what do they do ?	357
(A) Can you explain Report definition language (RDL) file in reporting services?	358
(B) What is the basic process of making a report in reporting services?.....	359
(B) How can we consume reports in ASP.NET?.....	360
(I) Can you explain the difference between private and shared data sources?	362
(A) How does reports caching in reporting services work ?.....	362
(I) What are the major differences between Crystal and SQL reporting services?	364
Chapter 20:- ASP.NET 2.0	364
(I) What improvements are provided in ASP.NET 2.0?.....	364
(I) How does ASP.NET 2.0 eliminate tedious coding?	365
(I) How do we encrypt web.config files in ASP.NET 2.0 ?	367
(A) With the above technique can you encrypt everything in the web.config file?	369
(A) In .NET 1.X how was the encryption implemented for config files?	370
(B) Can you explain membership and role providers in ASP.Net 2.0?	371
(I) What kind of security web controls are introduced in ASP.NET 2.0?	376
(I) Can you explain master pages concept in ASP.NET?	377
(I) what is the concept of Web parts?	380
(A) What are the different components of the web part framework?.....	381
(I) What are partial classes in ASP.NET ?.....	388
(I) Can you explain generics in .NET ?	388
(I) Can you explain the concept of generic collection?	389
Chapter 21:- How to	390
(B) How do you send a email using ASP.NET ?.....	390
(B) How did you deployment and setup in ASP.NET ?	391
Chapter 22:- .NET 3.5.....	393
(I) Define LINQ ?	394
(I) We already have common data access model what is special about LINQ?	395
(I) How can you make entity classes from the table itself ?	395
(A) How can we transform LINQ to objects ?.....	396
(A) How to transform LINQ to ADO.NET ?.....	396
(A) How to transform LINQ to SQL ?.....	396
(A) How to transform LINQ to XML ?	396
(A) How to transform LINQ to entities ?.....	396
(A) Can you explain Delegate Instantiation?.....	396
(A) Can you explain Anonymous methods ?.....	396
(A) What is Yield in LINQ ?	396
(A) Can you explain Lambda Expressions ?.....	396
(A) What are Instance methods and Extension methods ?	396
(A) What are Anonymous types ?.....	396
(A) Revision of Simple Query syntax for LINQ ?.....	396

(I) What is silver light?	396
---------------------------------	-----

What is View

-Uses of view

-Types of View

-View Syntax

-Alter/Modify View

-Rename View

-Drop View

1. Create City Table

```
CREATE TABLE City (  
    ID int primary key identity(1,1),  
    Name varchar(50)  
);
```

2. Insert Some sample Data in City Table

```
Insert into City(Name)VALUES('Mumbai');
```

```
Insert into City(Name)VALUES('Pune');
```

```
Insert into City(Name)VALUES('Hyderabad');
```

```
Insert into City(Name)VALUES('Banglore');
```

```
Insert into City(Name)VALUES('Chennai');
```

```
Insert into City(Name)VALUES('Delhi');
```

```
Select * from City
```

3. Create City Table

```
CREATE TABLE Employee (  
    EmpID int primary key identity(1,1),  
    Name varchar(50),  
    Department varchar(50),  
    Address varchar(50),  
    CityId int,  
    Mobile varchar(10),  
    Email varchar(30),
```

FOREIGN KEY (CityId) REFERENCES City(Id)

);

4. Insert Some sample Data in City Table

Insert into Employee(Name, Department, Address, CityId, Mobile, Email)

VALUES('Peter', '.Net', '12/5, Andheri', 1, '9876543210', 'peter@gmail.com');

Insert into Employee(Name, Department, Address, CityId, Mobile, Email)

VALUES('John', 'Java', '125A, Ameerpet', 3, '7876543210', 'john@gmail.com');

Insert into Employee(Name, Department, Address, CityId, Mobile, Email)

VALUES('Ram', 'UI', '105, Aundh', 2, '9870003210', 'ram@gmail.com');

Insert into Employee(Name, Department, Address, CityId, Mobile, Email)

VALUES('Shyam', 'Python', '18, Shyam Vihar', 1, '6576543210', 'shyan@gmail.com');

Insert into Employee(Name, Department, Address, CityId, Mobile, Email)

VALUES('Rohit', 'Java', '25C, XYZ', 5, '9006543210', 'rohit@gmail.com');

Insert into Employee(Name, Department, Address, CityId, Mobile, Email)

VALUES('Lucy', '.Net', '21/215, ABC, Aundh', 2, '9876543000', 'lucy@gmail.com')

Select * from Employee

What is View

A view is a virtual table whose contents are defined by a query.

A view is a database object that has no values.

It is a virtual table

A View does not contain any data itself

Uses of views

It prevents users from seeing specific columns and rows from tables.

Views are used to implement the security mechanism in SQL Server.

Views are generally used to focus, simplify, and customize the perception each user has of the database.

Types of views

1. User-Defined Views

Users define these views to meet their specific requirements.

It can also divide into two types one is the simple view, and another is the complex view.

2. System-Defined Views

System-defined views are predefined and existing views stored in SQL Server, such as Tempdb, Master, and temp

Each system views has its own properties and functions.

They can automatically attach to the user-defined databases.

Syntax to Create View

```
CREATE VIEW <schema_name>.<view_name>
```

```
AS
```

```
SELECT column1, column2, ...
```

```
FROM table1, table2,...
```

```
[WHERE];
```

-Alter/Modify View

```
ALTER VIEW <schema_name>.<view_name>
```

AS

SELECT column1, column2, ...

FROM table1, table2,...

[WHERE];

SELECT e.Name, e.Department, c.Name as City

FROM Employee e

Inner JOIN City c ON e.CityId=c.Id where c.Name='Pune';

-Rename View

SP_RENAME View_Old_Name, View_New_Name

-Drop View

DROP VIEW [IF EXISTS] schema_name.view_name;

Constraints in SQL

Constraints in SQL Server

Constraints are the predefined set of rules and restrictions applied on the tables or columns for restricting unauthorized values to be inserted into the tables.

This ensures the accuracy and reliability of the data in the database.

Constraints maintain the data integrity and accuracy in the table.

Categorization of Constraints

Table level constraints:

These constraints apply to the entire table that limit the types of data that can be entered into the table.

Its definitions are specified after creating the table using the ALTER statement.

Column level constraints:

These constraints apply to the single or multiple columns to limit the types of data that can be entered into the column.

Its definitions are specified while creating the table.

Types of Constraints

There are six types of constraints in SQL

Not Null Constraint

Default Constraint

Check Constraint

Unique Constraint

Primary Constraint

Foreign Constraint

1. Not Null Constraints

We can restrict NULL value from being inserted into a given column by using a NOT NULL constraint.

We can apply NOT NULL constraints either during the creation of the table or after creating the table using the ALTER statement.

If we are using a Not Null Constraint for a column then we cannot ignore the value of this column during an insert of data into the table.

Syntax

```
CREATE TABLE Table_Name
```

```
(
```

```
    Column_Name Datatype CONSTRAINT Constraint_Name NOT NULL,
```

```
);
```


Exp:

```
CREATE TABLE Employee
```

```
(
```

```
    Id int,
```

```
        Name varchar(20) CONSTRAINT NN_Const NOT NULL,
```

```
        Department varchar(20),
```

```
        Salary DECIMAL(10, 2)
```

```
);
```

Alter

```
ALTER TABLE Employee ALTER COLUMN Name varchar(20) not null;
```

```
Insert into Employee(Id, Department, Salary)VALUES(1, 'Development', 15000)
```

```
Insert into Employee(Id, Name, Department, Salary)VALUES(1, 'Peter', 'Development',  
15000)
```

2. Default Constraint

The Default constraint in SQL Server is used to fill the column with a default value.

Default constraints enable the SQL Server to insert a default value to a column when the user doesn't specify a value.

Note: IDENTITY and timestamp columns can't be associated with the default constraint.

Example

```
CREATE TABLE Employee
```

```
(
```

```
    Id int,
```

```
        Name varchar(20) CONSTRAINT NN_Const NOT NULL,
```

```
        Department varchar(20) CONSTRAINT Def_Const DEFAULT 'Develpoment',
```

```
        Salary DECIMAL(10, 2)
```

```
);
```

Alter

```
ALTER TABLE Employee ADD CONSTRAINT def_const DEFAULT 'Develpoment' for  
Department
```

```
Insert into Employee(Id, Name, Salary)VALUES(1, 'Peter', 15000)
```

3. Check Constraint

Check constraint checks for a specific condition before inserting data into a table.

It ensures that all the inserted values in a column must follow the specific rule.

It controls the value in a particular column and assures no corrupted information is entered in a column.

Example

```
CREATE TABLE Employee
```

```
(
```

```
    Id int,
```

```
    Name varchar(20) CONSTRAINT NN_Const NOT NULL,
```

```
    Department varchar(20) CONSTRAINT Def_Const DEFAULT 'Develpoment',
```

```
    Salary DECIMAL(10, 2) CONSTRAINT Check_Const CHECK(Salary>18000)
```

```
);
```

Alter

```
ALTER TABLE Employee ADD CONSTRAINT Check_Const  
CHECK(Salary>18000)
```

```
Insert into Employee(Id, Name, Salary)VALUES(1, 'Peter', 15000)
```

```
Insert into Employee(Id, Name, Salary)VALUES(1, 'Peter', 25000)
```

4. Unique Constraint

When you want a column or columns not to accept any duplicate values, then you need to apply UNIQUE Constraint to that column or columns.

The UNIQUE constraint ensures that no duplicate values can be inserted into a column or combination of columns that are not part of the PRIMARY KEY and are participating in the UNIQUE constraint.

A table can contain any number of UNIQUE constraints.

We can apply the UNIQUE constraint on any data type column such as integer, character, money, etc.

It is similar to the primary key, but it allows one null value.

Example

```
CREATE TABLE Employee
```

```
(
```

```
  Id int,
```

```
    Name varchar(20) CONSTRAINT NN_Const NOT NULL,
```

```
    Department varchar(20) CONSTRAINT Def_Const DEFAULT 'Develpoment',
```

```
    Salary DECIMAL(10, 2) CONSTRAINT Check_Const CHECK (Salary>18000),
```

```
    Mobile varchar(10) CONSTRAINT Unique_Key Unique
```

```
);
```

Alter

```
ALTER TABLE Employee ADD CONSTRAINT Unique_Key Unique(Mobile)
```

```
Insert into Employee(Id, Name, Mobile)VALUES(1, 'Peter', 9876543210)
```

```
Insert into Employee(Id, Name, Mobile)VALUES(2, 'John', 9876543210)
```

5. Primary Constraint

It is the combination of Not null and Unique Key constraints.

It always contains unique values in a column.

We cannot enter the null, empty, or duplicate values in the primary key constraints columns.

We mainly used this constraint to enforce the entity integrity of the table.

Example

```
CREATE TABLE Employee
```

```
(
```

```
    Id int CONSTRAINT Const_pk primary key,
```

```
    Name varchar(20) CONSTRAINT NN_Const NOT NULL,
```

```
    Department varchar(20) CONSTRAINT Def_Const DEFAULT 'Develpoment',
```

```
    Salary DECIMAL(10, 2) CONSTRAINT Check_Const CHECK (Salary>18000),
```

```
    Mobile varchar(10) CONSTRAINT Unique_Key Unique
```

```
);
```

Alter Note:- to add primary key constraint on a column using alter command, that column should be not null

```
ALTER TABLE Employee ADD CONSTRAINT Const_pk PRIMARY KEY (Name);
```

```
Insert into Employee(Id, Name, Mobile)VALUES(1, 'Peter', 9876543210)
```

```
Insert into Employee(Id, Name, Mobile)VALUES(1, 'John', 9876543255)
```

6. Foreign Constraint

A Foreign key creates a relation between two tables.

The first table contains a primary key and the second table contains a foreign key.

A foreign key is a database key that links two tables together.

foreign key column in one table refers to the primary key column of another table.

The common column that is present in both the tables need not have the same name but their data type must be the same.

```
CREATE TABLE Department
```

```
(
```

```
    Id int primary key identity(1,1),
```

```
    Name varchar(20) CONSTRAINT Unique_KeyDept Unique
```

```
);
```

```
Insert into Department(Name)VALUES('Development');
```

```
Insert into Department(Name)VALUES('Testing');
```

```
Insert into Department(Name)VALUES('HR');
```

```
Select * from Department
```

```
CREATE TABLE Employee
```

```
(
```

```
  Id int CONSTRAINT Const_pk primary key,
```

```
  Name varchar(20) CONSTRAINT NN_Const NOT NULL,
```

```
  DepartmentId int Constraint for_Key References Department(Id),
```

```
  Salary DECIMAL(10, 2) CONSTRAINT Check_Const CHECK (Salary>18000),
```

```
  Mobile varchar(10) CONSTRAINT Unique_Key Unique
```

```
);
```

```
Alter
```

```
  ALTER TABLE Employee ADD CONSTRAINT For_Key FOREIGN KEY  
(DepartmentId) REFERENCES Department(Id)
```

```
  Insert into Employee(Id, Name, DepartmentId, Mobile)VALUES(1, 'John', 1,  
9876543255)
```

Trigger in SQL Server

```
CREATE TRIGGER [schema].trigger_name
```

```
ON table_name
```

```
AFTER {INSERT, UPDATE, DELETE}
```

```
[NOT FOR REPLICATION]
```

```
AS
```

```
{SQL_Statements}
```

```
=====
```

```
Create table UserRegister(
```

```
ID int identity(1,1),
```

```
Name varchar(20),
```

```
Mobile varchar(10),
```

```
DOB DATE,
```

```
Department varchar(20),
```

```
Salary decimal(10,2)
```

```
)
```

```
Insert into
```

```
UserRegister(Name,Mobile,DOB,Department,Salary)VALUES('Peter','9876980000','07.10.1998',  
,'.Net',55000)
```

```
Select * from UserRegister
```


=====

Create table RegisterLog(

Id int identity(1,1),

UserId int not null,

Operation varchar(10),

UpdatedDate Date,

)

=====Trigger On Insert=====

CREATE TRIGGER trgUserRegisterInsert ON UserRegister

FOR INSERT

AS

Declare @Id int;

SELECT @Id = Ins.ID from inserted Ins;

INSERT INTO RegisterLog(UserId, Operation, UpdatedDate)values(@Id, 'INSERT',
GETDATE());

Select * from UserRegister;

Select * from RegisterLog

=====Trigger On Update=====

CREATE TRIGGER trgUserRegisterUpdate ON UserRegister

FOR UPDATE

AS

Declare @Id int;

```
SELECT @Id = Ins.ID from Inserted Ins;
```

```
INSERT INTO RegisterLog(UserId, Operation, UpdatedDate)values(@Id, 'Updatee',  
GETDATE());
```

```
=====DDL Trigger=====
```

```
CREATE TRIGGER tr_RestrictAlterTable
```

```
ON DATABASE
```

```
FOR ALTER_TABLE
```

```
AS
```

```
BEGIN
```

```
PRINT 'YOU CANNOT ALTER TABLES'
```

```
ROLLBACK TRANSACTION
```

```
END
```

```
--Exp
```

```
ALTER TABLE Student
```

```
ADD Test varchar(20);
```

```
=====Trigger On Delete=====
```

```
CREATE TRIGGER trgUserRegisterDelete ON UserRegister
```

```
FOR Delete
```

```
AS
```

```
Declare @Id int;
```

```
SELECT @Id = dl.ID from Deleted dl;
```

```
INSERT INTO RegisterLog(UserId, Operation, UpdatedDate)values(@Id, 'Delete',  
GETDATE());
```

=====Alter trigger=====

ALTER TRIGGER [dbo].[tr_UserRegisterUpdate] ON [dbo].[UserRegister]

FOR UPDATE

AS

Declare @Id int;

SELECT @Id = Ins.ID from Inserted Ins;

INSERT INTO RegisterLog(UserId, Operation, UpdatedDate)values(@Id, 'Update',
GETDATE());

=====Drop Trigger=====

DROP TRIGGER [IF EXISTS] [schema_name].[trigger1, trigger2, ...];

--Exp

DROP TRIGGER tr_UserRegisterUpdate;

=====Enable/Dissable Trigger=====

DISABLE TRIGGER [schema_name.][trigger_name]

ON [object_name | DATABASE | ALL SERVER]

ENABLE TRIGGER [schema_name.][trigger_name]

ON [object_name | DATABASE | ALL SERVER]

--Exp

ENABLE TRIGGER [dbo].[tr_UserRegisterInsert]

ON [dbo].[UserRegister]

=====Logon=====

```
Select * from sys.dm_exec_session
```

<https://docs.microsoft.com/en-us/sql/relational-databases/triggers/logon-triggers?view=sql-server-ver16>

Tasks

1.Create Form for registration using Ado.net ?

2. Write a POS application?

DBAccess file complete code || POS Application

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Data;
```

```
using System.Data.SqlClient;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
namespace WindowsFormsApplication1.DAL
```

```
{
```

```
    class DBAccess
```

```
    {
```

```
        private static SqlConnection objConnection;
```

```
        private static SqlDataAdapter objDataAdapter;
```

```
        public static string ConnectionString = @"Data
Source=DESKTOP-TA2ABCD\SQLEXPRESS; Initial Catalog=GautamPOS; User Id=sa;
Password=123456";
```

```
private static void OpenConnection()
```

```
{
```

```
    try
```

```
    {
```

```
        if (objConnection == null)
```

```
        {
```

```
            objConnection = new SqlConnection(ConnectionString);
```

```
            objConnection.Open();
```

```
        }
```

```
    else
```

```
    {
```

```
        if(objConnection.State!=ConnectionState.Open)
```

```
        {
```

```
            objConnection = new SqlConnection(ConnectionString);
```

```
        objConnection.Open();

    }

}

}

catch(Exception ex)

{

}

}

private static void CloseConnection()

{

    try

    {

        if(!(objConnection==null))

        {

            if(objConnection.State==System.Data.ConnectionState.Open)

            {

                objConnection.Close();
```



```
        objConnection.Dispose();

    }

}

}

catch {}

}
```

```
public static DataTable FillDataTable(string query, DataTable Table)

{

    try

    {

        OpenConnection();

        objDataAdapter = new SqlDataAdapter(query, objConnection);

        objDataAdapter.Fill(Table);

        objDataAdapter.Dispose();

        CloseConnection();

        return Table;

    }

}
```

```
}

catch(Exception ex)

{

    return null;

}

}

public static SqlDataReader ExecuteReader(string cmd)

{

    try

    {

        SqlDataReader objReader;

        objConnection = new SqlConnection(ConnectionString);

        OpenConnection();

        SqlCommand cmdRedr = new SqlCommand(cmd, objConnection);

        objReader = cmdRedr.ExecuteReader(CommandBehavior.CloseConnection);

        cmdRedr.Dispose();

    }

}
```

```
        return objReader;

    }

    catch

    {

        return null;

    }

}

public static bool ExecuteQuery(string query)

{

    try

    {

        using (SqlConnection connection = new SqlConnection(ConnectionString))

        {

            using (SqlCommand cmd = new SqlCommand(query, connection))

            {

                connection.Open();

                cmd.ExecuteNonQuery();

            }

        }

    }

}
```

```
        connection.Close();

        return true;

    }

}

}

catch (Exception ex)

{

    return false;

}

}

}

}
```

Angular is a robust front-end JavaScript framework that is widely used for app development. With the increased popularity, there is a high demand for Angular developers. This article on Angular Interview Questions will present some commonly asked questions and how to answer them. The questions are bifurcated into two levels, beginner and advanced.

Angular Interview Questions For Beginners

1. What is Angular? Why was it introduced?

Angular was introduced to create Single Page applications. This framework brings structure and consistency to web applications and provides excellent scalability and maintainability.

Angular is an open-source, JavaScript framework wholly written in TypeScript. It uses HTML's syntax to express your application's components clearly.

2. What is TypeScript?

TypeScript is a superset of JavaScript that offers excellent consistency. It is highly recommended, as it provides some syntactic sugar and makes the code base more comfortable to understand and maintain. Ultimately, TypeScript code compiles down to JavaScript that can run efficiently in any environment.

3. What is data binding? Which type of data binding does Angular deploy?

Data binding is a phenomenon that allows any internet user to manipulate Web page elements using a Web browser. It uses dynamic HTML and does not require complex scripting or programming. We use data binding in web pages that contain interactive components such as forms, calculators, tutorials, and games. Incremental display of a webpage makes data binding convenient when pages have an enormous amount of data.

Angular uses the two-way binding. Any changes made to the user interface are reflected in the corresponding model state. Conversely, any changes in the model state are reflected in the UI state. This allows the framework to connect the DOM to the Model data via the controller. However, this approach affects performance since every change in the DOM has to be tracked.

4. What are Single Page Applications (SPA)?

Single-page applications are web applications that load once with new features just being mere additions to the user interface. It does not load new HTML pages to display the new page's content, instead generated dynamically. This is made possible through JavaScript's ability to manipulate the DOM elements on the existing page itself. A SPA approach is faster, thus providing a seamless user experience.

5. Differentiate between Angular and AngularJS

The following table depicts the aspects of Angular vs AngularJS in detail:

Feature	AngularJS	Angular
Language	JavaScript	TypeScript
Architecture	Supports Model-View-Controller design	Uses components and directives
Mobile support	Not supported by mobile browsers	Supports all popular mobile browsers

Dependency Injection	Doesn't support	Supports
Routing	@routeProvider is used to provide routing information	@Route configuration is used to define routing information
Management	Difficult to manage with an increase in source code size	Better structured, easy to create and manage bigger applications

6. What are decorators in Angular?

Decorators are a design pattern or functions that define how Angular features work. They are used to make prior modifications to a class, service, or filter. Angular supports four types of decorators, they are:

1. Class Decorators
2. Property Decorators
3. Method Decorators

7. Mention some advantages of Angular.

Some of the common advantages of Angular are -

1. [MVC architecture](#) - Angular is a full-fledged MVC framework. It provides a firm opinion on how the application should be structured. It also offers bi-directional data flow and updates the real DOM.

2. Modules: Angular consists of different design patterns like components, directives, pipes, and services, which help in the smooth creation of applications.
3. Dependency injection: Components dependent on other components can be easily worked around using this feature.
4. Other generic advantages include clean and maintainable code, unit testing, reusable components, data binding, and excellent responsive experience.

Become job-ready and get complete job assistance by opting for the decade's hottest career option. Score your dream job in no time by enrolling in our Full Stack Java Developer Job Guarantee Program Today!

8. What are the new updates with Angular10?



Older versions of
TypeScript not supported



Warnings about
CommonJS imports



Optional strict setting

NGCC

Ngcc features



Compiler update



URL routing updation



Deprecated APIs



Bug fixes



New default browser
configuration

- Older versions of TypeScript not supported - Previous versions of Angular supported typescript 3.6, 3.7, and even 3.8. But with Angular 10, TypeScript bumped to TypeScript 3.9.
- Warnings about CommonJS imports - Logging of unknown property bindings or element names in templates is increased to the "error" level, which was previously a "warning" before.

- Optional strict setting - Version 10 offers a stricter project setup when you create a new workspace with ng new command.

ng new --strict

NGCC Feature - Addition of NGCC features with a program based entry point finder.

- Updated URL routing
- Deprecated APIs - Angular 10 has several deprecated APIs.
- Bug fixes - With this Angular 10 version, there have been a number of bug fixes, important ones being the compiler avoiding undefined expressions and the core avoiding a migration error when a nonexistent symbol is imported.
- New Default Browser Configuration - Browser configuration for new projects has been upgraded to outdo older and less used browsers.

9. What are Templates in Angular?

Angular Templates are written with HTML that contains Angular-specific elements and attributes. In combination with the model and controller's information, these templates are further rendered to provide a dynamic view to the user.

10. What are Annotations in Angular?

Annotations in Angular are used for creating an annotation array. They are the metadata set on the class that is used to reflect the Metadata library.

11. What are Directives in Angular?

Directives are attributes that allow the user to write new HTML syntax specific to their applications. They execute whenever the Angular compiler finds them in the DOM. Angular supports three types of directives.

1. Component Directives
2. Structural Directives
3. Attribute Directives

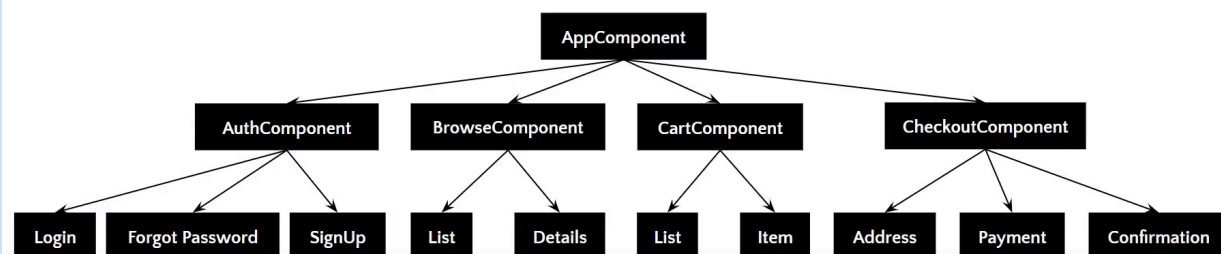
12. What is an AOT compilation? What are its advantages?

The Ahead-of-time (AOT) compiler converts the Angular HTML and TypeScript code into JavaScript code during the build phase, i.e., before the browser downloads and runs the code.

Some of its advantages are as follows.

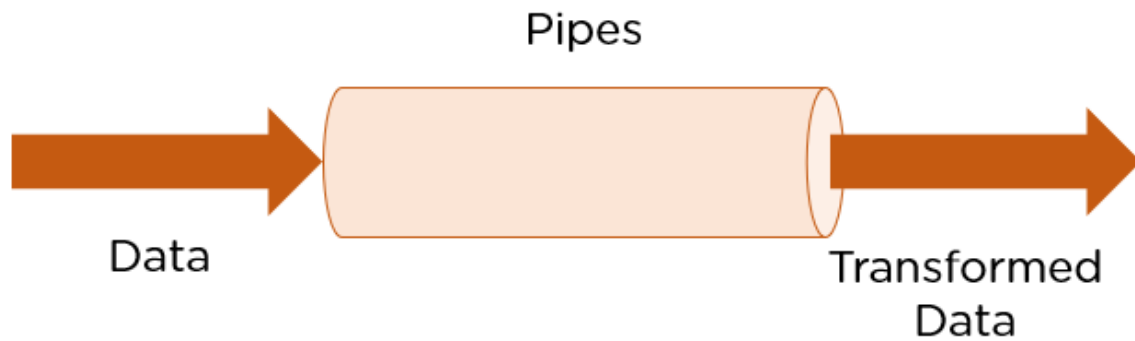
1. Faster rendering
2. Fewer asynchronous requests
3. Smaller Angular framework download size
4. Quick detection of template errors
5. Better security

13. What are Components in Angular?



Components are the basic building blocks of the user interface in an Angular application. Every component is associated with a template and is a subset of directives. An Angular application typically consists of a root component, which is the AppComponent, that then branches out into other components creating a hierarchy.

14. What are Pipes in Angular?



Pipes are simple functions designed to accept an input value, process, and return as an output, a transformed value in a more technical understanding. Angular supports several built-in pipes. However, you can also create custom pipes that cater to your needs.

Some key features include:

1. Pipes are defined using the pipe “|” symbol.
2. Pipes can be chained with other pipes.
3. Pipes can be provided with arguments by using the colon (:) sign.

15. What is the PipeTransform interface?

As the name suggests, the interface receives an input value and transforms it into the desired format with a `transform()` method. It is typically used to implement custom pipes.

```
import { Pipe, PipeTransform } from '@angular/core';
```

```
@Pipe({
```

```
  name: 'demopipe'
```

```
})
```

```
export class DemopipePipe implements PipeTransform {  
  
  transform(value: unknown, ...args: unknown[]): unknown {  
  
    return null;  
  
  }  
  
}
```

16. What are Pure Pipes?

These pipes are pipes that use pure functions. As a result of this, a pure pipe doesn't use any internal state, and the output remains the same as long as the parameters passed stay the same. Angular calls the pipe only when it detects a change in the parameters being passed. A single instance of the pure pipe is used throughout all components.

17. What are Impure Pipes?

For every change detection cycle in Angular, an impure pipe is called regardless of the change in the input fields. Multiple pipe instances are created for these pipes. Inputs passed to these pipes can be mutable.

By default, all pipes are pure. However, you can specify impure pipes using the pure property, as shown below.

```
@Pipe({  
  
  name: 'demopipe',  
  
  pure : true/false
```

```
}}
```

```
export class DemopipePipe implements PipeTransform {
```

18. What is an NgModule?

NgModules are containers that reserve a block of code to an application domain or a workflow. @NgModule takes a metadata object that generally describes the way to compile the template of a component and to generate an injector at runtime. In addition, it identifies the module's components, directives, and pipes, making some of them public, through the export property so that external components can use them.

19. What are filters in Angular? Name a few of them.

Filters are used to format an expression and present it to the user. They can be used in view templates, controllers, or services. Some inbuilt filters are as follows.

- date - Format a date to a specified format.
- filter - Select a subset of items from an array.
- Json - Format an object to a JSON string.
- limitTo - Limits an array/string, into a specified number of elements/characters.
- lowercase - Format a string to lowercase.

20. What is view encapsulation in Angular?

View encapsulation defines whether the template and styles defined within the component can affect the whole application or vice versa. Angular provides three encapsulation strategies:

1. Emulated - styles from the main HTML propagate to the component.
2. Native - styles from the main HTML do not propagate to the component.
3. None - styles from the component propagate back to the main HTML and therefore are visible to all components on the page.

21. What are controllers?

AngularJS controllers control the data of AngularJS applications. They are regular JavaScript Objects. The ng-controller directive defines the application controller.

22. What do you understand by scope in Angular?

The scope in Angular binds the HTML, i.e., the view, and the JavaScript, i.e., the controller. It as expected is an object with the available methods and properties. The scope is available for both the view and the controller. When you make a controller in Angular, you pass the \$scope object as an argument.

23. Explain the lifecycle hooks in Angular

In Angular, every component has a lifecycle. Angular creates and renders these components and also destroys them before removing them from the DOM. This is achieved with the help of lifecycle hooks. Here's the list of them -

1. `ngOnChanges()` - Responds when Angular sets/resets data-bound input properties.
2. `ngOnInit()` - Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties/
3. `ngDoCheck()` - Detect and act upon changes that Angular can't or won't detect on its own.
4. `ngAfterContentInit()` - Responds after Angular projects external content into the component's view.
5. `ngAfterContentChecked()` - Respond after Angular checks the content projected into the component.
6. `ngAfterViewInit()` - Respond after Angular initializes the component's views and child views.
7. `ngAfterViewChecked()` - Respond after Angular checks the component's views and child views.
8. `ngOnDestroy` - Cleanup just before Angular destroys the directive/component.

24. What is String Interpolation in Angular?

String Interpolation is a one-way data-binding technique that outputs the data from TypeScript code to HTML view. It is denoted using double curly braces. This template expression helps display the data from the component to the view.

```
{{ data }}
```

25. What are Template statements?

Template statements are properties or methods used in HTML for responding to user events. With these template statements, the application that you create or are working on, can have the capability to engage users through actions such as submitting forms and displaying dynamic content.

For example,

```
<button (click)="deleteHero()">Delete hero</button>
```

The template here is deleteHero. The method is called when the user clicks on the button.

Angular Interview Questions For Advanced Level

26. What is the difference between AOT and JIT?

Ahead of Time (AOT) compilation converts your code during the build time before the browser downloads and runs that code. This ensures faster rendering to the browser. To specify AOT compilation, include the `--aot` option with the `ng build` or `ng serve` command.

The Just-in-Time (JIT) compilation process is a way of compiling computer code to machine code during execution or run time. It is also known as dynamic compilation. JIT compilation is the default when you run the `ng build` or `ng serve` CLI commands.

27. Explain the @Component Decorator.

TypeScript class is one that is used to create components. This genre of class is then decorated with the `"@Component"` decorator. The decorator's purpose is to accept a metadata object that provides relevant information about the component.

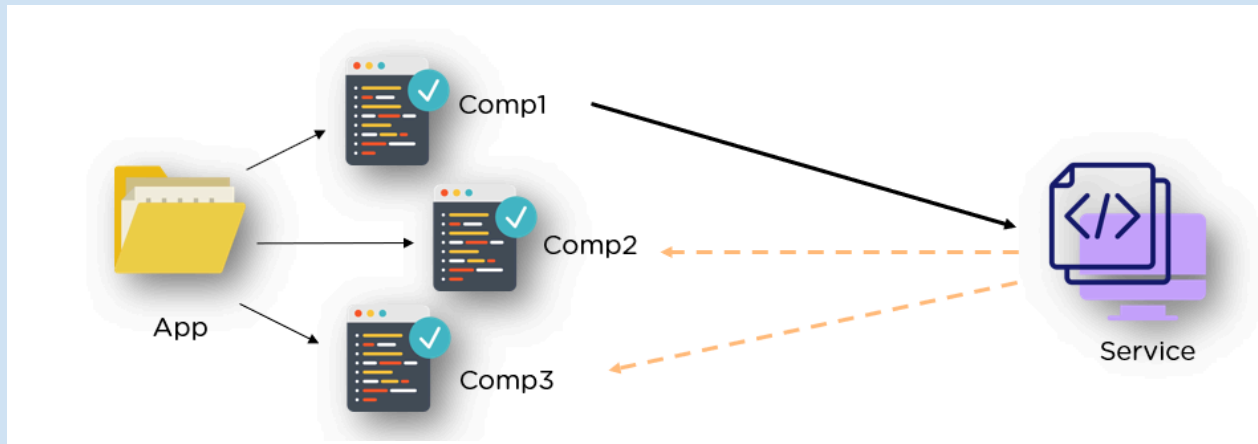
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'example';
}
```

The image above shows an App component - a pure TypeScript class decorated with the “@Component” decorator. The metadata object that gets accepted by the decorator provides properties like templateUrl, selector, and others, where the templateUrl property points to an HTML file defining what you see on the application.

28. What are Services in Angular?

[Angular Services](#) perform tasks that are used by multiple components. These tasks could be data and image fetching, network connections, and database management among others. They perform all the operational tasks for the components and avoid rewriting of code. A service can be written once and injected into all the components that use that service.



29. What are Promises and Observables in Angular?

While both the concepts deal with Asynchronous events in Angular, Promises handle one such event at a time while observables handle a sequence of events over some time.

Promises - They emit a single value at a time. They execute immediately after creation and are not cancellable. They are Push errors to the child promises.

Observables - They are only executed when subscribed to them using the `subscribe()` method. They emit multiple values over a period of time. They help perform operations like `forEach`, `filter`, and `retry`, among others. They deliver errors to the subscribers. When the `unsubscribe()` method is called, the listener stops receiving further values.

30. What is `ngOnInit`? How is it defined?

`ngOnInit` is a lifecycle hook and a callback method that is run by Angular to indicate that a component has been created. It takes no parameters and returns a void type.

```
export class MyComponent implements OnInit {
```

```
  constructor() {}
```

```
  ngOnInit(): void {
```

```
//....
```

```
}
```

```
}
```

31. How to use ngFor in a tag?

The ngFor directive is used to build lists and tables in the HTML templates. In simple terms, this directive is used to iterate over an array or an object and create a template for each element.

```
<ul>
```

```
<li *ngFor = "let items in itemlist"> {{ item }} </li>
```

```
</ul>
```

1. "Let item" creates a local variable that will be available in the template
2. "Of items" indicates that we are iterating over the items iterable.
3. The * before ngFor creates a parent template.

32. What are Template and Reactive forms?

Template-driven approach

- In this method, the conventional form tag is used to create forms. Angular automatically interprets and creates a form object representation for the tag.
- Controls can be added to the form using the NGModel tag. Multiple controls can be grouped using the NGControlGroup module.
- A form value can be generated using the "form.value" object. Form data is exported as JSON values when the submit method is called.
- Basic HTML validations can be used to validate the form fields. In the case of custom validations, directives can be used.
- Arguably, this method is the simplest way to create an Angular App.

Reactive Form Approach

- This approach is the programming paradigm oriented around data flows and propagation of change.
- With Reactive forms, the component directly manages the data flows between the form controls and the data models.
- Reactive forms are code-driven, unlike the template-driven approach.
- Reactive forms break from the traditional declarative approach.
- Reactive forms eliminate the anti-pattern of updating the data model via two-way data binding.
- Typically, Reactive form control creation is synchronous and can be unit tested with synchronous programming techniques.

33. What is Bootstrap? How is it embedded into Angular?

Bootstrap is a powerful toolkit. It is a collection of HTML, CSS, and JavaScript tools for creating and building responsive web pages and web applications.

There are two ways to embed the bootstrap library into your application.

1. Angular Bootstrap via CDN - Bootstrap CDN is a public Content Delivery Network. It enables you to load the CSS and JavaScript files remotely from its servers.
2. Angular Bootstrap via NPM - Another way to add Bootstrap to your Angular project is to install it into your project folder by using NPM (Node Package Manager).

```
npm install bootstrap
```

```
npm install jquery
```

34. What is Eager and Lazy loading?

Eager loading is the default module-loading strategy. Feature modules under Eager loading are loaded before the application starts. This is typically used for small size applications.

Lazy loading dynamically loads the feature modules when there's a demand. This makes the application faster. It is used for bigger applications where all the modules are not required at the start of the application.

35. What type of DOM does Angular implement?

DOM (Document Object Model) treats an XML or HTML document as a tree structure in which each node is an object representing a part of the document.

Angular uses the regular DOM. This updates the entire tree structure of HTML tags until it reaches the data to be updated. However, to ensure that the speed and performance are not affected, Angular implements Change Detection.

With this, you have reached the end of the article. We highly recommend brushing up on the core concepts for an interview. It's always an added advantage to write the code in places necessary.

36. Why were client-side frameworks like Angular introduced?

Client-side frameworks like Angular were introduced to provide a more responsive user experience. By using a framework, developers can create web applications that are more interactive and therefore provide a better user experience.

Frameworks like Angular also make it easier for developers to create single-page applications (SPAs). SPAs are web applications that only need to load a single HTML page. This makes them much faster and more responsive than traditional web applications.

Overall, client-side frameworks like Angular were introduced in order to improve the user experience of web applications. By making web applications more responsive and easier to develop, they provide a better experience for both developers and users.

37. How does an Angular application work?

An Angular application is a Single Page Application, or SPA. This means that the entire application lives within a single page, and all of the resources (HTML, CSS, JavaScript,

etc.) are loaded when the page is first loaded. Angular uses the Model-View-Controller, or MVC, architecture pattern to manage its data and views. The Model is the data that the application uses, the View is what the user sees, and the Controller is responsible for managing communication between the Model and the View.

When a user interacts with an Angular application, the Angular framework will automatically update the View to reflect any changes in the data. This means that Angular applications are very responsive and fast, as the user does not need to wait for the page to reload in order to see updated data.

Angular applications are also very scalable, as they can be divided into small modules that can be loaded independently of each other. This means that an Angular application can be easily extended with new functionality without having to rewrite the entire application.

Overall, Angular applications are very fast, responsive, and scalable. They are easy to develop and extend, and provide a great user experience.

The following is an example of coding from an angular.json file:

```
"build": {  
  
  "builder": "@angular-devkit/build-angular:browser",  
  
  "options": {  
  
    "outputPath": "dist/angular-starter",  
  
    "index": "src/index.html",  
  
    "main": "src/main.ts",  
  
    "polyfills": "src/polyfills.ts",  
  
    "tsConfig": "tsconfig.app.json",
```

```
"aot": false,  
  
"assets": [  
  
  "src/favicon.ico",  
  
  "src/assets"  
  
],  
  
"styles": [  
  
  "./node_modules/@angular/material/prebuilt-themes/deeppurple-amber.css",  
  
  "src/style.css"  
  
]  
  
}  
  
}
```

38. Explain components, modules and services in Angular.

Components, modules and services are the three fundamental building blocks in Angular. Components are the smallest, self-contained units in an Angular application. They are typically used to represent a view or UI element, such as a button or a form input field.

Code example:

```
import { Component, OnInit } from '@angular/core';  
  
@Component({
```

```
selector: 'app-test',
```

```
templateUrl: './test.component.html',
```

```
styleUrls: ['./test.component.css']
```

```
))
```

```
export class TestComponent implements OnInit {
```

```
  constructor() {}
```

```
  ngOnInit() {
```

```
  }
```

```
}
```

Modules are larger units that group together one or more related components. Services are singleton objects that provide specific functionality throughout an Angular application, such as data access or logging.

Code example:

```
import { BrowserModule } from '@angular/platform-browser';
```

```
import { NgModule } from '@angular/core';
```

```
import { AppComponent } from './app.component';
```

```
import { TestComponent } from './test/text.component';
```

```
@NgModule({
```

```
  declarations: [
```

```
    AppComponent,  
  
    TestComponent  
  
  ],  
  
  imports: [  
  
    BrowserModule  
  
  ],  
  
  providers: [],  
  
  bootstrap: [AppComponent]  
  
  })  
  
export class AppModule { }
```

Each component in Angular has its own isolated scope. This means that a component's dependencies (services, other components, etc.) are not accessible to any other component outside of its own scope. This isolation is important for ensuring modularity and flexibility in an Angular application.

Services, on the other hand, are not isolated and can be injected into any other unit in an Angular application (component, module, service, etc.). This makes them ideal for sharing data or functionality across the entire app.

Code example:

```
import { Injectable } from '@angular/core';  
  
@Injectable({  
  
  providedIn: 'root'
```



```
})
```

```
export class TestServiceService {
```

```
  constructor() {}
```

```
}
```

When designing an Angular application, it is important to keep these three building blocks in mind. Components should be small and self-contained, modules should group together related components, and services should provide shared functionality across the entire app. By following this design principle, you can create an Angular application that is modular, flexible, and easy to maintain.

39. How are Angular expressions different from JavaScript expressions?

One major difference between Angular expressions and JavaScript expressions is that Angular expressions are compiled while JavaScript expressions are not. This means that Angular expressions are more efficient since they're already pre-processed. Additionally, Angular expressions can access scope properties while JavaScript expressions cannot. Finally, Angular expressions support some additional features such as filters and directives which aren't available in JavaScript expressions.

Javascript expression example:

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>JavaScript Test</title>
```

```
</head>
```

```
<body>
```

```
<div id="foo"><div>
```

```
</div>
```

```
<script>
```

```
'use strict';
```

```
let bar = {};
```

```
document.getElementById('foo').innerHTML = bar.x;
```

```
</script>
```

```
</html>
```

Angular expression example:

```
import { Component, OnInit } from '@angular/core';
```

```
@Component({
```

```
  selector: 'app-new',
```

```
  template: `
```

```
<h4>{{message}}</h4>
```

```
`
```

```
  styleUrls: ['./new.component.css']
```

```
}}
```

```
export class NewComponent implements OnInit {
```

```
  message:object = {};
```

```
  constructor() { }
```

```
  ngOnInit() {
```

```
  }
```

```
}
```

40. Angular by default, uses client-side rendering for its applications.

This means that the Angular application is rendered on the client-side — in the user's web browser. Client-side rendering has a number of advantages, including improved performance and better security. However, there are some drawbacks to using client-side rendering, as well. One of the biggest drawbacks is that it can make your application more difficult to debug.

Client-side rendering is typically used for applications that are not heavily data-driven. If your application relies on a lot of data from a server, client-side rendering can be very slow. Additionally, if you're using client-side rendering, it's important to be careful about how you load and cache your data. If you're not careful, you can easily end up with an application that is very slow and difficult to use. When rendered on the server-side, this is called Angular Universal.

41. How do you share data between components in Angular?

Sharing data between components in Angular is simple and easy. To share data, all you need to do is use the Angular CLI to generate a new service. This service can be injected into any component and will allow the components to share data.

To generate a new service, use the following Angular CLI command:

ng generate service my-data-service

This will create a new service file called my-data-service.ts in the src/app folder.

Inject the service into any component that needs to share data:

```
import { MyDataService } from './my-data.service';  
  
constructor(private myDataService: MyDataService) { }
```

Once injected, the service will be available in the component as this.myDataService.

To share data between components, simply use the setData() and getData() methods:

```
this.myDataService.setData('some data');  
  
const data = this.myDataService.getData();
```

42. Explain the concept of dependency injection.

Dependency injection is a technique used to create loosely coupled code. It allows pieces of code to be reused without the need for hard-coded dependencies. This makes code more maintainable and easier to test. Dependency injection is often used in frameworks like AngularJS, ReactJS, and VueJS. It is also possible to use dependency injection in vanilla JavaScript. To use dependency injection in JavaScript, you need a dependency injection library. There are many different libraries available, but they all work in basically the same way.

The first step is to create a dependency injection container. This is a simple object that will hold all of the dependencies that your code needs. Next, you need to register all of the dependencies that your code will need with the container. The registration process will vary depending on the library you are using, but it is usually just a matter of providing the dependency's name and constructor function.

Once all of the dependencies have been registered, you can then inject them into your code. The injection process will again vary depending on the library you are using, but it

is usually just a matter of providing the dependency's name. The library will then take care of instantiating the dependency and passing it to your code.

Dependency injection can be a great way to make your code more modular and easier to maintain. It can also make it easier to unit test your code, as you can inject mock dependencies instead of the real ones. If you are using a framework that supports dependency injection, then it is probably already being used in your code. If you are not using a framework, then you can still use dependency injection by choosing a library and following the steps outlined above.

Code example:

```
import { Injectable } from '@angular/core';
```

```
  @Injectable({
```

```
    providedIn: 'root'
```

```
  })
```

```
  export class TestService {
```

```
    importantValue:number = 42;
```

```
    constructor() { }
```

```
    returnImportantValue(){
```

```
      return this.importantValue;
```

```
    }
```

```
  }
```

The injectable dependencies are created after adding the @Injectable decorator to a class. The dependency above is then injected into the following component:

```
import { TestService } from '../test.service';

import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-test',

  templateUrl: './test.component.html',

  styleUrls: ['./test.component.css']
})

export class TestComponent implements OnInit {

  value:number;

  constructor(private testService:TestService) {}

  ngOnInit() {

    this.value = this.testService.returnImportantValue();

  }

}
```

43. Explain MVVM architecture.

MVVM architecture is an architectural pattern used mainly in software engineering. It stands for Model-View-ViewModel. MVVM is a variation of the traditional MVC (Model-View-Controller) software design pattern. The main difference between the two is that MVVM separates the user interface logic from the business logic, while MVC

separates the data access logic from the business logic. This separation of concerns makes it easier to develop, test, and maintain software applications.

The Model layer in MVVM architecture is responsible for storing and managing data. It can be a database, a web service, or a local data source. The View layer is responsible for displaying data to the user. It can be a graphical user interface (GUI), a command-line interface (CLI), or a web page. The ViewModel layer is responsible for handling user input and updating the View layer accordingly. It contains the business logic of the application.

MVVM architecture is often used in conjunction with other software design patterns, such as Model-View-Presenter (MVP) and Model-View-Controller (MVC). These patterns can be used together to create a complete software application.

MVVM architecture is a popular choice for modern software applications. It allows developers to create applications that are more responsive and easier to maintain. Additionally, MVVM architecture can be used to create applications that can be easily ported to different platforms.

ASP.NET CORE INTERVIEW QUESTIONS

What is the ASP.NET Core?

ASP.NET Core is not an upgraded version of ASP.NET. ASP.NET Core is completely rewriting that work with the .net Core framework. It is much faster, configurable, modular, scalable, extensible, and has cross-platform support. It can work with both .NET Core and .net framework via the .NET standard framework. It is best suitable for developing cloud-based such as web applications, mobile applications, and IoT applications.

ASP.NET Core was primarily designed to make the most important part of the ASP.NET components under the concept learn and the compose framework where the previous ASP.NET components were released under a variety of different licenses periodically, The ASP.NET Core framework is a completely open-sourced framework. Apart from the other parts of the framework of the .NET framework libraries, the ASP.NET Core is primarily designed from scratch to be the platform-agnostic that performs seamlessly. It will allow the ASP.NET Core apps to be deployed on the various platforms or the o/s such as the macOS or Linux-based servers or certain devices.

What are the features provided by ASP.NET Core?

Following are the core features that are provided by the ASP.NET Core

- Built-in supports for
- Dependency Injection
- Built-in supports for the logging framework and it can be extensible
- Introduced a new, fast and cross-platform web server - Kestrel. So, a web application can run without IIS, Apache, and Nginx.
- Multiple hosting ways are supported
- It supports modularity, so the developer needs to include the module required by the application. However, the **.NET Core** framework is also providing the meta package that includes the libraries

- Command-line supports to creating, building, and running of the application
- There is no web.config file. We can store the custom configuration into an appsettings.json file
- There is no Global.asax file. We can now register and use the services in the startup class
- It has good support for asynchronous programming
- Support WebSocket and SignalR
- Provide protection against CSRF (Cross-Site Request Forgery)

What are the advantages of ASP.NET Core over ASP.NET?

There are the following advantages of ASP.NET Core over ASP.NET :

- It is cross-platform, so it can be run on Windows, Linux, and Mac.
- There is no dependency on framework installation because all the required dependencies are shipped with our application
- ASP.NET Core can handle more requests than the ASP.NET
- Multiple deployment options available with ASP.NET Core

What are Metapackages?

The framework .NET Core 2.0 introduced Metapackage which includes all the supported packages by ASP.NET code with their dependencies into one package. It helps us to do fast development as we don't require to include the individual ASP.NET Core packages. The assembly Microsoft.AspNetCore.All is a meta package provided by ASP.NET core.

In other words, the Metapackages of .NET Core describes the set of packages that are used together and acts as a parent of the child grouping structure. The Metapackages are referenced just like any other NuGet package naming convention such as "NETStandard.Library". An by referencing the meta-package, you have, then all its child packages will be having the reference of its dependent packages accordingly.

Can ASP.NET Core application work with full .NET 4.x Framework?

Yes. ASP.NET core application works with full .NET framework via the .NET standard library.

What is the startup class in ASP.NET core?

The startup class is the entry point of the ASP.NET Core application. Every .NET Core application must have this class. This class contains the application configuration related items. It is not necessary that the class name must be "Startup", it can be anything, we can configure the startup class in the Program class.

```
public class Program
```

```
{
```

```
    public static void Main(string[] args)
```

```
{
```

```
        CreateWebHostBuilder(args).Build().Run();
```

```
}
```

```
    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
```

```
        WebHost.CreateDefaultBuilder(args)
```

```
            .UseStartup<TestClass>();
```

```
}
```

What is the use of the ConfigureServices method of the startup class?

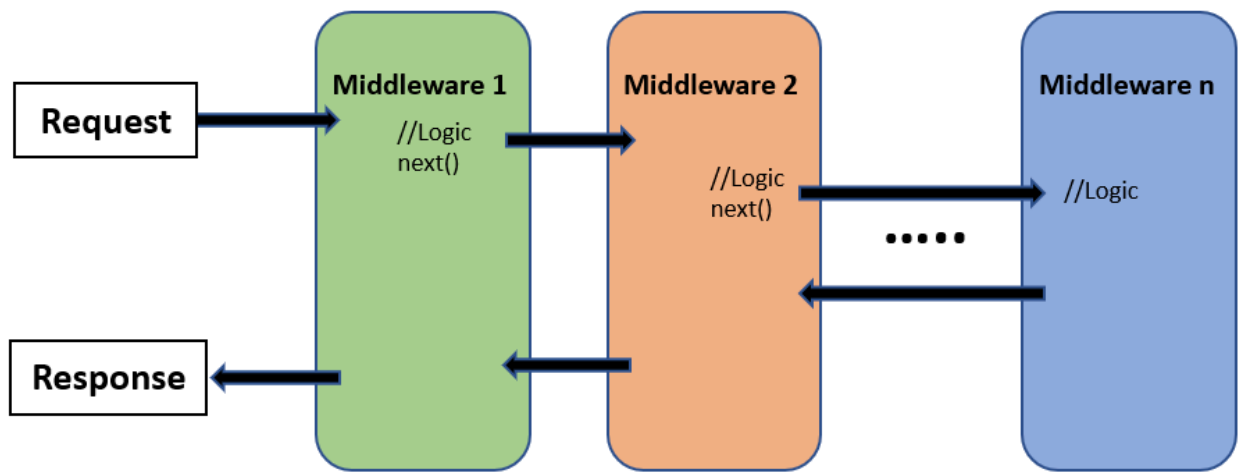
This is an optional method of startup class. It can be used to configure the services that are used by the application. This method calls first when the application is requested for the first time. Using this method, we can add the services to the DI container, so services are available as a dependency in the controller constructor.

What is the use of the Configure method of the startup class?

It defines how the application will respond to each HTTP request. We can configure the request pipeline by configuring the middleware. It accepts `IApplicationBuilder` as a parameter and also it has two optional parameters: `IHostingEnvironment` and `ILoggerFactory`. Using this method, we can configure built-in [middleware](#) such as routing, authentication, session, etc. as well as third-party middleware.

What is middleware?

It is software that is injected into the application pipeline to handle requests and responses. They are just like chained to each other and form as a pipeline. The incoming requests are passed through this pipeline where all middleware is configured, and middleware can perform some action on the request before passing it to the next middleware. Same as for the responses, they are also passing through the middleware but in reverse order.



While working with the ASP.NET Core framework, there are tons of built-in Middleware components available that are already made available that we can use directly that act as a plug and play components. If we don't want to use any of the in-built middleware, then we can also create our own Middleware components in asp.net core applications whenever we want. The most important point that you need to keep in mind is, that in ASP.NET Core a given Middleware component should only have a specific purpose which means it should be used for a single responsibility.

What is the difference between `ApplicationBuilder.Use()` and `ApplicationBuilder.Run()`?

We can use both the methods in Configure methods of the startup class. Both are used to add middleware delegates to the application request pipeline. The middleware adds using `ApplicationBuilder.Use` may call the next middleware in the pipeline whereas the middleware adds using `ApplicationBuilder.Run` method never calls the subsequent middleware. After `ApplicationBuilder.Run` method, system stop adding middleware in the request pipeline.

What is the use of the "Map" extension while adding middleware to the ASP.NET Core pipeline?

It is used for branching the pipeline. It branches the ASP.NET Core pipeline based on request path matching. If the request path starts with the given path, middleware on to that branch will execute.

```
public void Configure(IApplicationBuilder app)
{
    app.Map("/path1", Middleware1);

    app.Map("/path2", Middleware2);
}
```

What is routing in ASP.NET Core?

Routing is functionality that map incoming request to the route handler. The route can have values (extract them from the URL) that are used to process the request. Using the route, routing can find a route handler based on the URL. All the routes are registered when the application is started. There are two types of routing supported by ASP.NET Core

- The conventional routing
- Attribute routing

The Routing uses routes to map incoming requests with the route handler and Generates URL that is used in response. Mostly, the application has a single collection of routes and this collection is used for the process of the request. The

RouteAsync method is used to map incoming requests (that match the URL) with available in route collection.

How to enable Session in ASP.NET Core?

The middleware for the session is provided by the package `Microsoft.AspNetCore.Session`. To use the session in the ASP.NET Core application, we need to add this package to the csproj file and add the Session middleware to the ASP.NET Core request pipeline.

```
public class Startup

{

    public void ConfigureServices(IServiceCollection services)

    {

        ....

        ....

        services.AddSession();

        services.AddMvc();

    }

    public void Configure(IApplicationBuilder app, IHostingEnvironment env)

    {

        ....

        ....

    }

}
```

```
app.UseSession();
```

```
....
```

```
....
```

```
}
```

```
}
```

What are the various JSON files available in ASP.NET Core?

There are the following JSON files in ASP.NET Core :

- global.json
- launchsettings.json
- appsettings.json
- bundleconfig.json
- bower.json
- package.json

What is tag helper in ASP.NET Core?

It is a feature provided by the Razor view engine that enables us to write server-side code to create and render the HTML element in view (Razor). The tag-helper is a C# class that is used to generate the view by adding the HTML element. The functionality of the tag helper is very similar to the HTML helper of ASP.NET MVC.

Example:

```
//HTML Helper
```

```
@Html.TextBoxFor(model => model.FirstName, new { @class = "form-control",  
placeholder = "Enter Your First Name" })
```

```
//content with tag helper
```

```
<input asp-for="FirstName" placeholder="Enter Your First Name"  
class="form-control" />
```

```
//Equivalent HTML
```

```
<input placeholder="Enter Your First Name" class="form-control" id="FirstName"  
name="FirstName" value="" type="text">
```

How to disable Tag Helper at the element level?

We can disable Tag Helper at the element level using the opt-out character ("!"). This character must apply to open and close the Html tag.

Example

```
<!span asp-validation-for="phone" class="divPhone"></!span>
```

What are Razor Pages in ASP.NET Core?

This is a new feature introduced in ASP.NET Core 2.0. It follows a page-centric development model just like ASP.NET web forms. It supports all the features of ASP.NET Core.

Example

```
@page
```



```
<h1> Hello, Book Reader!</h1>
```

```
<h2> This is Razor Pages </h2>
```

The Razor pages start with the @page directive. This directive handle request directly without passing through the controller. The Razor pages may have code behind files, but it is not really a code-behind file. It is a class inherited from PageModel class.

How can we do the automatic model binding in Razor pages?

The Razor pages provide the option to bind property automatically when posting the data using the BindProperty attribute. By default, it only binds the properties only with non-GET verbs. we need to set SupportsGet property to true to bind a property on getting a request.

Example

```
public class Test1Model : PageModel
```

```
{
```

```
[BindProperty]
```

```
public string Name { get; set; }
```

```
}
```

How can we inject the service dependency into the controller?

There are three easy steps to add a custom service as a dependency on the controller.

Step 1: Create the service

```
public interface IHelloWorldService
```

```
{
```

```
    string SaysHello();
```

```
}
```

```
public class HelloWorldService: IHelloWorldService
```

```
{
```

```
    public string SaysHello()
```

```
    {
```

```
        return "Hello ";
```

```
    }
```

```
}
```

Step 2: Add this service to the Service container (service can either be added by singleton, transient, or scoped)

```
public void ConfigureServices(IServiceCollection services)
```

```
{
```

```
    ....
```

```
...
```

```
services.AddTransient<IHelloWorldService, HelloWorldService>();
```

```
...
```

```
...
```

```
}
```

Step 3: Use this service as a dependency in the controller

```
public class HomeController: Controller
```

```
{
```

```
    IHelloWorldService _helloWorldService;
```

```
    public HomeController(IHelloWorldService helloWorldService)
```

```
{
```

```
        _helloWorldService = helloWorldService;
```

```
}
```

```
}
```

How to specify the service life for a registered service that is added as a dependency?

ASP.NET Core allows us to specify the lifetime for registered services. The service

instance gets disposed of automatically based on a specified lifetime. So, we do not care about the cleaning these dependencies, it will take care of the ASP.NET Core framework. There are three types of lifetimes.

Singleton

ASP.NET Core will create and share a single instance of the service through the application life. The service can be added as a singleton using the `AddSingleton` method of `IServiceCollection`. ASP.NET Core creates a service instance at the time of registration and subsequent requests use this service instance. Here, we do not require to implement the Singleton design pattern and single instance maintained by the ASP.NET Core itself.

Example

```
services.AddSingleton<IHelloWorldService, HelloWorldService>();
```

Transient

ASP.NET Core will create and share an instance of the service every time to the application when we ask for it. The service can be added as Transient using the `AddTransient` method of `IServiceCollection`. This lifetime can be used in stateless service. It is a way to add lightweight service.

Example

```
services.AddTransient<IHelloWorldService, HelloWorldService>();
```

Scoped

ASP.NET Core will create and share an instance of the service per request to the application. It means that a single instance of service is available per request. It will create a new instance in the new request. The service can be added as scoped using an `AddScoped` method of `IServiceCollection`. We need to take care while the service registered via Scoped in middleware and inject the service in the `Invoke` or `InvokeAsync` methods. If we inject dependency via the constructor, it behaves like a singleton object.

```
services.AddScoped<IHelloWorldService, HelloWorldService>();
```

MVC Interview Questions

If you're planning to attend a .NET Interview, you may also be prepared for ASP.NET MVC interview questions. MVC is the framework used to build Web applications for .NET and C#. In this article, I list the top 50 MVC questions and their answers. The answers are code examples written by authors of C# Corner.

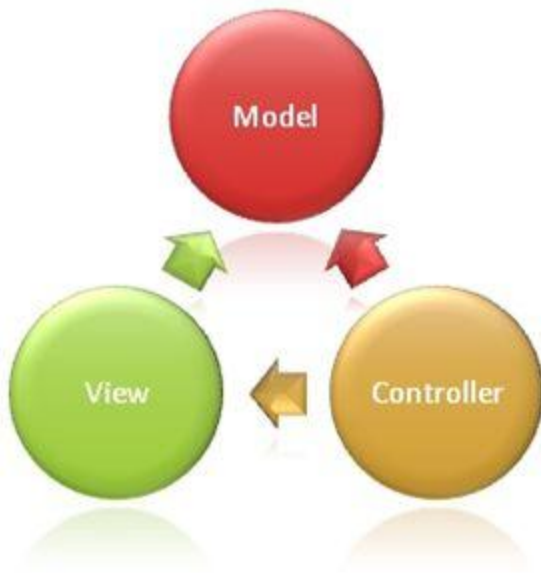
1. What is MVC (Model View Controller)?

Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representation of information from the way that information is presented to or accepted from the user.

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

- The Model represents the application core (for instance a list of database records).
- The View displays the data (the database records).
- The Controller handles the input (to the database records).

The MVC model also provides full control over HTML, CSS, and JavaScript.



The MVC model defines web applications with 3 logic layers,

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

The Model is the part of the application that handles the logic for the application data.

Often model objects retrieve data (and store data) from a database.

The View is the part of the application that handles the display of the data.

Most often the views are created from the model data.

The Controller is the part of the application that handles user interaction.

Typically controllers read data from a view, control user input, and send input data to the model.

The MVC separation helps you manage complex applications because you can focus on one aspect a time. For example, you can focus on the view without depending on the business logic. It also makes it easier to test an application.

The MVC separation also simplifies group development. Different developers can work on the view, the controller logic, and the business logic in parallel.

Learn more about ASP.NET MVC here: [Overview Of ASP.NET MVC](#)

2. What are the advantages of MVC?

Multiple view support

Due to the separation of the model from the view, the user interface can display multiple views of the same data at the same time.

Change Accommodation

User interfaces tend to change more frequently than business rules (different colors, fonts, screen layouts, and levels of support for new devices such as cell phones or PDAs) because the model does not depend on the views, adding new types of views to the system generally does not affect the model. As a result, the scope of change is confined to the view.

SoC – Separation of Concerns

Separation of Concerns is one of the core advantages of ASP.NET MVC. The MVC framework provides a clean separation of the UI, Business Logic, Model or Data.

More Control

The ASP.NET MVC framework provides more control over HTML, JavaScript, and CSS than the traditional Web Forms.

Testability

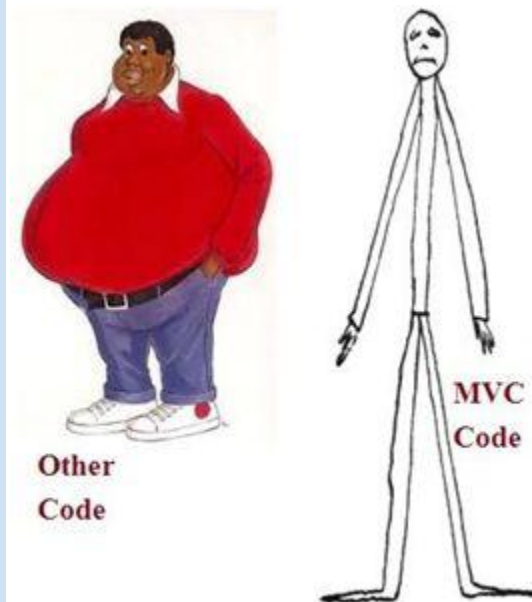
ASP.NET MVC framework provides better testability of the Web Application and good support for test driven development too.

Lightweight

ASP.NET MVC framework doesn't use View State and thus reduces the bandwidth of the requests to an extent.

Full features of ASP.NET

One of the key advantages of using ASP.NET MVC is that it is built on top of the ASP.NET framework and hence most of the features of the ASP.NET like membership providers, roles, etc can still be used.



Here is a detailed article on [Creating a Simple Application Using MVC 4.0](#).

3. Explain MVC application life cycle?

Any web application has two main execution steps, first understanding the request and depending on the type of the request sending out an appropriate response. MVC application life cycle is not different it has two main phases, first creating the request object and second sending our response to the browser.

Creating the request object,

The request object creation has four major steps. The following is a detailed explanation of the same.

Step 1 - Fill route

MVC requests are mapped to route tables which in turn specify which controller and action to be invoked. So if the request is the first request the first thing is to fill the route table with routes collection. This filling of the route table happens in the global.asax file

Step 2 - Fetch route

Depending on the URL sent "UrlRoutingModule" searches the route table to create "RouteData" object which has the details of which controller and action to invoke.

Step 3 - Request context created

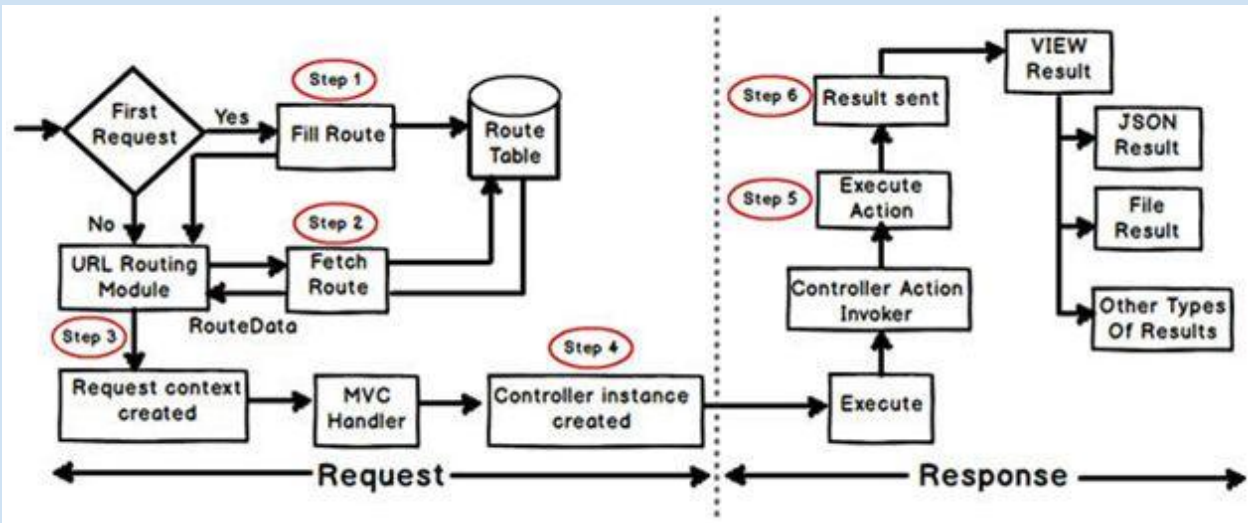
The "RouteData" object is used to create the "RequestContext" object.

Step 4 - Controller instance created

This request object is sent to "MvcHandler" instance to create the controller class instance. Once the controller class object is created it calls the "Execute" method of the controller class.

Creating a Response object

This phase has two steps executing the action and finally sending the response as a result to the view.



Here is a complete article on [ASP.Net MVC Life Cycle](#).

4. List out different return types of a controller action method?

There are total of nine return types we can use to return results from the controller to view.

The base type of all these result types is ActionResult.

ViewResult (View)

This return type is used to return a webpage from an action method.

PartialviewResult (Partialview)

This return type is used to send a part of a view that will be rendered in another view.

RedirectResult (Redirect)

This return type is used to redirect to any other controller and action method depending on the URL.

RedirectToRouteResult (RedirectToAction, RedirectToRoute)

This return type is used when we want to redirect to any other action method.

ContentResult (Content)

This return type is used to return HTTP content type like text/plain as the result of the action.

jsonResult (json)

This return type is used when we want to return a JSON message.

javascriptResult (javascript)

This return type is used to return JavaScript code that will run in the browser.

FileResult (File)

This return type is used to send binary output in response.

EmptyResult

This return type is used to return nothing (void) in the result.

Here is a complete article on [Various Return Types From MVC Controller](#).

5. What are the Filters in MVC?

In MVC, controllers define action methods and these action methods generally have a one-to-one relationship with UI controls such as clicking a button or a link, etc. For example, in one of our previous examples, the UserController class contained methods UserAdd, UserDelete, etc.

But many times we would like to perform some action before or after a particular operation. For achieving this functionality, ASP.NET MVC provides a feature to add pre and post-action behaviors on the controller's action methods.

Types of Filters

ASP.NET MVC framework supports the following action filters,

- *Action Filters*
Action filters are used to implement logic that gets executed before and after a controller action executes. We will look at Action Filters in detail in this chapter.
- *Authorization Filters*
Authorization filters are used to implement authentication and authorization for controller actions.
- *Result Filters*
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*
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 can also use exception filters to log errors.

Action filters are one of the most commonly used filters to perform additional data processing, or manipulating the return values or canceling the execution of an action or modifying the view structure at run time.

Here is a complete article on [Understanding Filters in MVC](#).

6. What are Action Filters in MVC?

Answer - Action Filters

Action Filters are additional attributes that can be applied to either a controller section or the entire controller to modify the way in which action is executed. These attributes are special .NET classes derived from System.Attribute which can be attached to classes, methods, properties, and fields.

ASP.NET MVC provides the following action filters,

- *Output Cache*
This action filter caches the output of a controller action for a specified amount of time.

- *Handle Error*
This action filter handles errors raised when a controller action executes.
- *Authorize*
This action filter enables you to restrict access to a particular user or role.

Now we will see the code example to apply these filters on an example controller ActionFilterDemoController. (ActionFilterDemoController is just used as an example. You can use these filters on any of your controllers.)

Output Cache

Code Example

Specifies the return value to be cached for 10 seconds.

```
public class ActionFilterDemoController: Controller { [HttpGet]  
OutputCache(Duration = 10)] public string Index() { return  
DateTime.Now.ToString("T"); } }
```

C#Copy

Learn more here: [ASP.NET MVC with Action Filters](#)

7. Explain what is routing in MVC? What are the three segments for routing important?

Routing is a mechanism to process the incoming URL that is more descriptive and gives the desired response. In this case, URL is not mapped to specific files or folder as was the case of earlier days web sites.

There are two types of routing (after the introduction of ASP.NET MVC 5).

1. Convention-based routing - to define this type of routing, we call MapRoute method and set its unique name, URL pattern and specify some default values.
2. Attribute-based routing - to define this type of routing, we specify the Route attribute in the action method of the controller.

Routing is the URL pattern that is mapped together to a handler, routing is responsible for incoming browser request for particular MVC controller. In other ways let us say routing help you to define a URL structure and map the URL with controller. There are three segments for routing that are important,

1. ControllerName
2. ActionMethodName
3. Parameter

Code Example

ControllerName/ActionMethodName/{ParamerName} and also route map coding written in a Global.asax file.

Learn more here - [Routing in MVC](#).

8. What is Route in MVC? What is Default Route in MVC?

A route is a URL pattern that is mapped to a handler. The handler can be a physical file, such as a .aspx file in a Web Forms application. A handler can also be a class that processes the request, such as a controller in an MVC application. To define a route, you create an instance of the [Route](#) class by specifying the URL pattern, the handler, and optionally a name for the route.

You add the route to the application by adding the Route object to the static Routes property of the RouteTable class. The Routes property is a RouteCollection object that stores all the routes for the application.

You typically do not have to write code to add routes in an MVC application. Visual Studio project templates for MVC include preconfigured URL routes. These are defined in the MVC Application class, which is defined in the Global.asax file.

Route definition	Example of matching URL
{controller}/{action}/{id}	/Products/show/beverages
{table}/Details.aspx	/Products/Details.aspx
blog/{action}/{entry}	/blog/show/123

{reporttype}/{year}/{month}/{day}	/sales/2008/1/5
{locale}/{action}	/US/show
{language}-{country}/{action}	/en-US/show

Default Route

The default ASP.NET MVC project templates add a generic route that uses the following URL convention to break the URL for a given request into three named segments.

URL: "{controller}/{action}/{id}"

This route pattern is registered via a call to the MapRoute() extension method of RouteCollection.

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index",
                        id = UrlParameter.Optional }
    );
}
```

Diagram annotations:

- An arrow points from the text "Ignore routes that end with axd extension" to the `{resource}.axd` part of the `IgnoreRoute` call.
- An arrow points from the text "Route Name" to the `name: "Default"` parameter.
- An arrow points from the text "URL Pattern" to the `url: "{controller}/{action}/{id}"` parameter.
- An arrow points from the text "Default Values" to the `defaults` parameter.

9. Mention what is the difference between TempData, View, and ViewBag?

In ASP.NET MVC there are three ways to pass/store data between the controllers and views.

ViewData

1. ViewData is used to pass data from controller to view.
2. It is derived from ViewDataDictionary class.
3. It is available for the current request only.
4. Requires typecasting for complex data types and checks for null values to avoid an error.

5. If redirection occurs, then its value becomes null.

ViewBag

1. ViewBag is also used to pass data from the controller to the respective view.
2. ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0
3. It is also available for the current request only.
4. If redirection occurs, then its value becomes null.
5. It doesn't require typecasting for the complex data type.

TempData

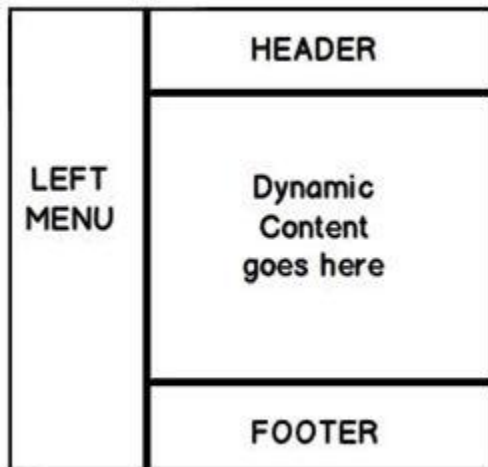
1. TempData is derived from TempDataDictionary class
2. TempData is used to pass data from the current request to the next request
3. It keeps the information for the time of an HTTP Request. This means only from one page to another. It helps to maintain the data when we move from one controller to another controller or from one action to another action
4. It requires typecasting for complex data types and checks for null values to avoid an error. Generally, it is used to store only one time messages like the error messages and validation messages

Learn more here: [Difference Between ViewData, ViewBag, and TempData](#)

10. What is Partial View in MVC?

A partial view is a chunk of HTML that can be safely inserted into an existing DOM. Most commonly, partial views are used to componentize Razor views and make them easier to build and update. Partial views can also be returned directly from controller methods. In this case, the browser still receives text/html content but not necessarily HTML content that makes up an entire page. As a result, if a URL that returns a partial view is directly invoked from the address bar of a browser, an incomplete page may be displayed. This may be something like a page that misses title, script and style sheets. However, when the same URL is invoked via a script, and the response is used to insert HTML within the existing DOM, then the net effect for the end-user may be much better and nicer.

Partial view is a reusable view (like a user control) which can be embedded inside another view. For example, let's say all the pages of your site have a standard structure with left menu, header, and footer as in the following image,



```

@model IEnumerable<MvcApplications8.Models.Product>

@{
    ViewBag.Title = "Index";
}


<h2>Products</h2>

<table>
    <tr>
        <th>
            Name
        </th>
        <th>
            Description
        </th>
        <th>
            Price
        </th>
        <th>
            EDD
        </th>
    </tr>

    @foreach (var item in Model) {
        <tr>
            <td>
                @item.Name
            </td>
            <td>
                @item.Description
            </td>
            <td>
                @(item.Price/10)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id=item.ProductId }) |
                @Html.ActionLink("Details", "Details", new { id=item.ProductId }) |
                @Html.ActionLink("Delete", "Delete", new { id=item.ProductId })
            </td>
        </tr>
    }
</table>

```

Let's make this piece of code reusable



Learn more here - [Partial View in MVC](#)

11. Explain what is the difference between View and Partial View?

View

- It contains the layout page.
- Before any view is rendered, viewstart page is rendered.
- A view might have markup tags like body, HTML, head, title, meta etc.
- The view is not lightweight as compare to Partial View.

Partial View

- It does not contain the layout page.
- Partial view does not verify for a viewstart.cshtml. We cannot put common code for a partial view within the viewStart.cshtml.page.
- Partial view is designed specially to render within the view and just because of that it does not consist any mark up.
- We can pass a regular view to the RenderPartial method.

Learn more here - [Partial View in MVC](#)

12. What are HTML helpers in MVC?

With MVC, HTML helpers are much like traditional ASP.NET Web Form controls.

Just like web form controls in ASP.NET, HTML helpers are used to modify HTML. But HTML helpers are more lightweight. Unlike Web Form controls, an HTML helper does not have an event model and a view state.

In most cases, an HTML helper is just a method that returns a string.

With MVC, you can create your own helpers, or use the built in HTML helpers.

Standard HTML Helpers

HTML Links

The easiest way to render an HTML link in is to use the `Html.ActionLink()` helper. With MVC, the `Html.ActionLink()` does not link to a view. It creates a link to a controller action.

ASP Syntax

```
<%=Html.ActionLink("About this Website", "About")%>
```

ASP.NET (C#)Copy

The first parameter is the link text, and the second parameter is the name of the controller action.

The `Html.ActionLink()` helper above, outputs the following HTML:

```
<a href="/Home/About">About this Website</a>
```

MarkupCopy

The `Html.ActionLink()` helper has several properties:

- Property Description.
- `.linkText` The link text (label).
- `.actionName` The target action.
- `.routeValues` The values passed to the action.
- `.controllerName` The target controller.
- `.htmlAttributes` The set of attributes to the link.
- `.protocol` The link protocol.
- `.hostname` The host name for the link.
- `.fragment` The anchor target for the link.

HTML Form Elements

There following HTML helpers can be used to render (modify and output) HTML form elements:

- `BeginForm()`
- `EndForm()`
- `TextArea()`
- `TextBox()`
- `CheckBox()`
- `RadioButton()`
- `ListBox()`
- `DropDownList()`
- `Hidden()`
- `Password()`

ASP.NET Syntax C#

```
<%= Html.ValidationSummary("Create was unsuccessful. Please correct  
the errors and try again.") %> <% using (Html.BeginForm()) { %> <p>  
<label for="FirstName">First Name:</label> <%=
```

```

Html.TextBox("FirstName") %> <%= Html.ValidationMessage("FirstName",
"") %> </p> <p> <label for="LastName">Last Name:</label> <%=
Html.TextBox("LastName") %> <%= Html.ValidationMessage("LastName",
"") %> </p> <p> <label for="Password">Password:</label> <%=
Html.Password("Password") %> <%= Html.ValidationMessage("Password",
"") %> </p> <p> <label for="Password">Confirm Password:</label> <%=
Html.Password("ConfirmPassword") %> <%=
Html.ValidationMessage("ConfirmPassword", " ") %> </p> <p> <label
for="Profile">Profile:</label> <%= Html.TextArea("Profile", new
{cols=60, rows=10})%> </p> <p> <%= Html.CheckBox("ReceiveNewsletter")
%> <label for="ReceiveNewsletter" style="display:inline">Receive
Newsletter</label> </p> <p> <input type="submit" value="Register" />
</p> <%}%>

```

ASP.NET (C#)Copy

Learn more here - [HTML Helpers in MVC: Part 1](#)

13. Explain attribute based routing in MVC?

In ASP.NET MVC 5.0 we have a new attribute route, by using the "Route" attribute we can define the URL structure. For example in the below code we have decorated the "GotoAbout" action with the route attribute. The route attribute says that the "GotoAbout" can be invoked using the URL structure "Users/about".

Hide Copy Code

```

public class HomeController: Controller { [Route("Users/about")]
public ActionResult GotoAbout() { return View(); } }

```

C#Copy

Learn more here - [Attribute Based Routing in ASP.Net MVC 5](#)

14. What is TempData in MVC?

TempData is a dictionary object to store data temporarily. It is a TempDataDictionary class type and instance property of the Controller base class.

TempData is able to keep data for the duration of a HTTP request, in other words it can keep live data between two consecutive HTTP requests. It will help us to pass the state between action methods. TempData only works with the current and subsequent request. TempData uses a session variable to store the data. TempData Requires type casting when used to retrieve data.

TempDataDictionary is inherited from the IDictionary<string, object>, ICollection<KeyValuePair<string, object>>, IEnumerable<KeyValuePair<string, object>> and IEnumerable interfaces.

Example

```
public ActionResult FirstRequest() { List < string > TempDataTest =
new List < string > (); TempDataTest.Add("Tejas");
TempDataTest.Add("Jignesh"); TempDataTest.Add("Rakesh");
TempData["EmpName"] = TempDataTest; return View(); } public
ActionResult ConsecutiveRequest() { List < string > modelData =
TempData["EmpName"] as List < string > ; TempData.Keep(); return
View(modelData); }
```

C#Copy

Learn more here - [All About the TempData in MVC](#)

15. What is Razor in MVC?

ASP.NET MVC has always supported the concept of "view engines" - which are the pluggable modules that implement different template syntax options. The "default" view engine for ASP.NET MVC uses the same .aspx/.ascx/. master file templates as ASP.NET Web Forms. Other popular ASP.NET MVC view engines are Spart&Nhaml.

MVC 3 has introduced a new view engine called Razor.

Why is Razor?

1. Compact & Expressive.
2. Razor minimizes the number of characters and keystrokes required in a file, and enables a fast coding workflow. Unlike most template syntaxes, you do not need to interrupt your coding to explicitly denote server blocks within your HTML. The parser is smart enough to infer this from your code. This enables a really compact and expressive syntax which is clean, fast and fun to type.
3. Easy to Learn: Razor is easy to learn and enables you to quickly be productive with a minimum of effort. We can use all your existing language and HTML skills.
4. Works with any Text Editor: Razor doesn't require a specific tool and enables you to be productive in any plain old text editor (notepad works great).
5. Has great Intellisense:
6. Unit Testable: The new view engine implementation will support the ability to unit test views (without requiring a controller or web-server, and can be hosted in any unit test project - no special app-domain required).

Learn more here - [Brief Introduction to MVC3](#)

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

Razor View Engine VS ASPX View Engine

Razor View Engine	ASPX View Engine (Web form view engine)
The namespace used by the Razor View Engine is System.Web.Razor	The namespace used by the ASPX View Engine is System.Web.Mvc.WebFormViewEngine
The file extensions used by the Razor View Engine are different from a web form view engine. It uses cshtml with C# and vbhtml with vb for views, partial view, templates and layout pages.	The file extensions used by the Web Form View Engines are like ASP.Net web forms. It uses the ASPX extension to view the aspc extension for partial views or User Controls or templates and master extensions for layout/master pages.

The Razor View Engine is an advanced view engine that was introduced with MVC 3.0. This is not a new language but it is markup.	A web form view engine is the default view engine and available from the beginning of MVC
Razor has a syntax that is very compact and helps us to reduce typing.	The web form view engine has syntax that is the same as an ASP.Net forms application.
The Razor View Engine uses @ to render server-side content.	The ASPX/web form view engine uses "<%= %>" or "<%: %>" to render server-side content.
By default all text from an @ expression is HTML encoded.	There is a different syntax ("<%: %>") to make text HTML encoded.
Razor does not require the code block to be closed, the Razor View Engine parses itself and it is able to decide at runtime which is a content element and which is a code element.	A web form view engine requires the code block to be closed properly otherwise it throws a runtime exception.
The Razor View Engine prevents Cross-Site Scripting (XSS) attacks by encoding the script or HTML tags before rendering to the view.	A web form View engine does not prevent Cross-Site Scripting (XSS) attacks.
The Razor Engine supports Test Driven Development (TDD).	Web Form view engine does not support Test Driven Development (TDD) because it depends on the System.Web.UI.Page class to make the testing complex.
Razor uses "@* â€¦ *" for multiline comments.	The ASPX View Engine uses "<!--...-->" for markup and "/* â€¦ */" for C# code.
There are only three transition characters with the Razor View Engine.	There are only three transition characters with the Razor View Engine.

The Razor View Engine is a bit slower than the ASPX View Engine.

Conclusion

Razor provides a new view engine with a streamlined code for focused templating. Razor's syntax is very compact and improves the readability of the markup and code. By default, MVC supports ASPX (web forms) and Razor View Engine. MVC also

supports third-party view engines like Spark, Nhaml, NDjango, SharpDOM and so on. ASP.NET MVC is open source.

Learn more here - [ASPX View Engine VS Razor View Engine](#)

17. What are the Main Razor Syntax Rules?

Answer

- Razor code blocks are enclosed in @{ ... }
- Inline expressions (variables and functions) start with @
- Code statements end with semicolon
- Variables are declared with the var keyword
- Strings are enclosed with quotation marks
- C# code is case sensitive
- C# files have the extension .cshtml

C# Example

```
<!-- Single statement block --> @ { varmyMessage = "Hello World"; }  
<!-- Inline expression or variable --> < p > The value of myMessage  
is: @myMessage < /p> <!-- Multi-statement block --> @ { var greeting =  
"Welcome to our site!"; varweekDay = DateTime.Now.DayOfWeek;  
vargreetingMessage = greeting + " Here in Huston it is: " + weekDay; }  
< p > The greeting is: @greetingMessage < /p>
```

C#Copy

Learn more here - [Introduction to Microsoft ASP.NET MVC 3 Razor View Engine](#)

18. How do you implement Forms authentication in MVC?

Answer

Authentication is giving access to the user for a specific service by verifying his/her identity using his/her credentials like username and password or email and password. It assures that the correct user is authenticated or logged in for a specific service and the

right service has been provided to the specific user based on their role that is nothing but authorization.

ASP.NET forms authentication occurs after IIS authentication is completed. You can configure forms authentication by using forms element with in web.config file of your application. The default attribute values for forms authentication are shown below,

```
<system.web> <authenticationmode="Forms"> <formsloginUrl="Login.aspx"
protection="All" timeout="30" name=".ASPXAUTH" path="/"
requireSSL="false" slidingExpiration="true" defaultUrl="default.aspx"
cookieless="UseDeviceProfile" enableCrossAppRedirects="false" />
</authentication> </system.web>
```

MarkupCopy

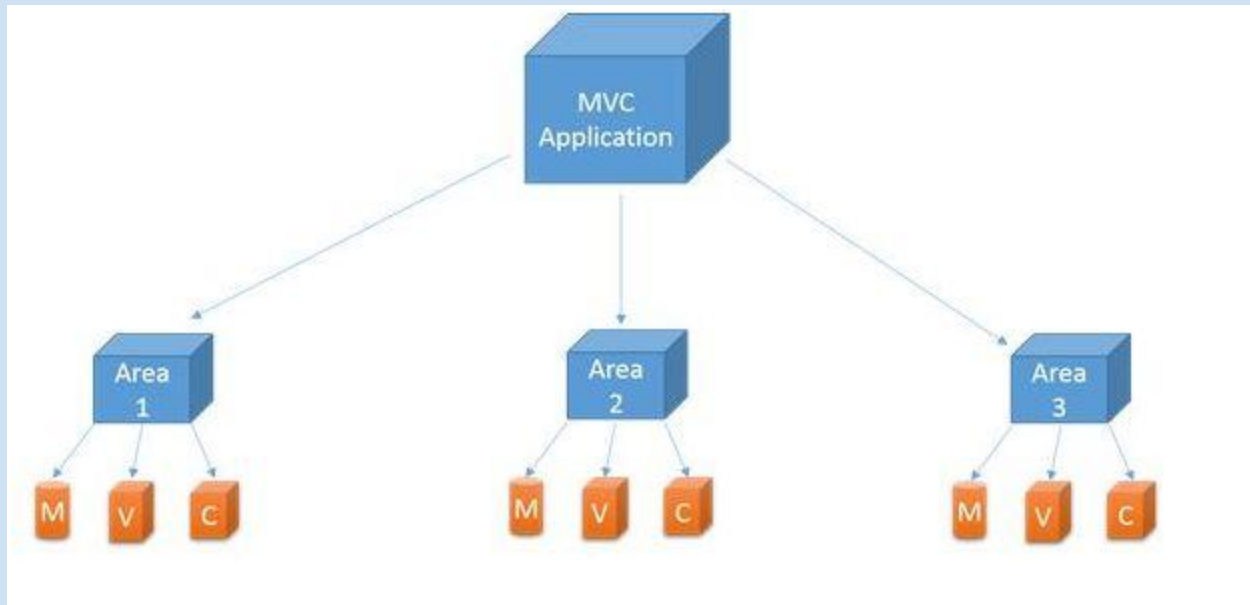
The FormsAuthentication class creates the authentication cookie automatically when SetAuthCookie() or RedirectFromLoginPage() methods are called. The value of authentication cookie contains a string representation of the encrypted and signed FormsAuthenticationTicket object.

Learn more here - [Form Authentication in MVC 5: Part 1](#)

19. Explain Areas in MVC?

Answer

From ASP.Net MVC 2.0 Microsoft provided a new feature in MVC applications, Areas. Areas are just a way to divide or “isolate” the modules of large applications in multiple or separated MVC. like,



When you add an area to a project, a route for the area is defined in an `AreaRegistration` file. The route sends requests to the area based on the request URL. To register routes for areas, you add code to the `Global.asax` file that can automatically find the area routes in the `AreaRegistration` file.

```
AreaRegistration.RegisterAllAreas();
```

Benefits of Area in MVC

1. Allows us to organize models, views and controllers into separate functional sections of the application, such as administration, billing, customer support and much more.
2. Easy to integrate with other Areas created by another.
3. Easy for unit testing.

Learn more here - [What Are Areas in ASP.Net MVC - Part 6](#)

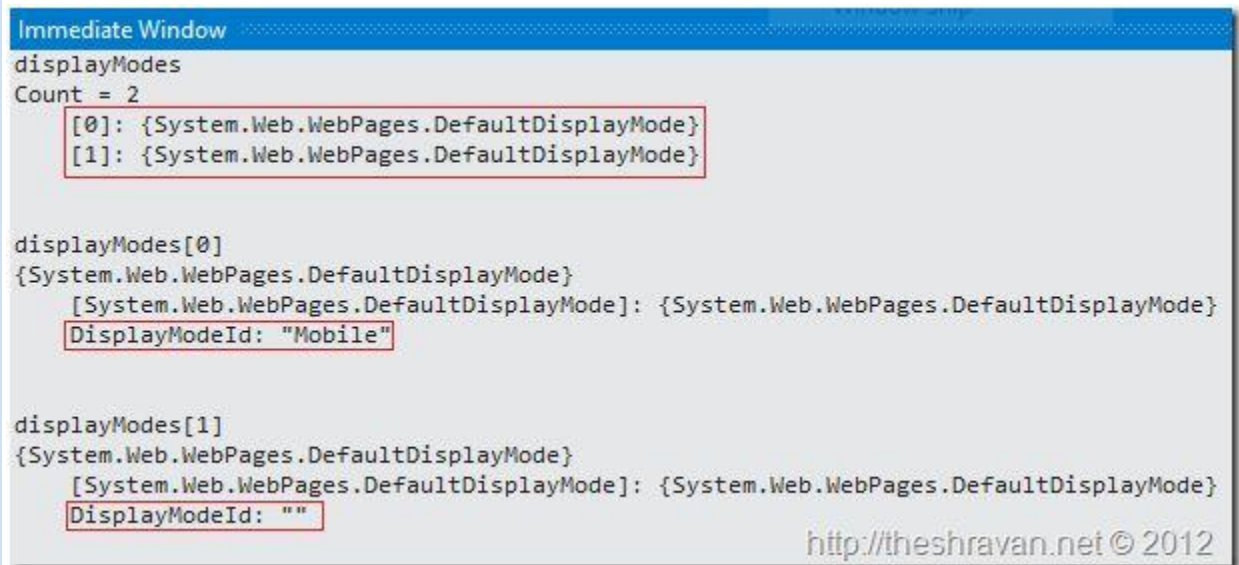
20. Explain the need of display mode in MVC?

Answer

DisplayModes give you another level of flexibility on top of the default capabilities we saw in the last section. DisplayModes can also be used along with the previous feature so we will simply build off of the site we just created.

Using display modes involves in 2 steps

1. We should register Display Mode with a suffix for particular browser using "DefaultDisplayMode" class in Application_Start() method in the Global.asax file.
2. View name for particular browser should be appended with suffix mentioned in first step.



The screenshot shows the 'Immediate Window' in Visual Studio. It displays the following information:

```
displayModes
Count = 2
[0]: {System.Web.WebPages.DefaultDisplayMode}
[1]: {System.Web.WebPages.DefaultDisplayMode}

displayModes[0]
{System.Web.WebPages.DefaultDisplayMode}
[System.Web.WebPages.DefaultDisplayMode]: {System.Web.WebPages.DefaultDisplayMode}
DisplayModeId: "Mobile"

displayModes[1]
{System.Web.WebPages.DefaultDisplayMode}
[System.Web.WebPages.DefaultDisplayMode]: {System.Web.WebPages.DefaultDisplayMode}
DisplayModeId: ""
```

The 'DisplayModeId' for the first mode is 'Mobile', and for the second mode, it is an empty string. The URL <http://theshravan.net> © 2012 is visible in the bottom right corner.

1. Desktop browsers (without any suffix. e.g.: Index.cshtml, _Layout.cshtml).
 2. Mobile browsers (with a suffix "Mobile". e.g.: Index.Mobile.cshtml, Layout.Mobile.cshtml)
- If you want design different pages for different mobile device browsers (any different browsers) and render them depending on the browser requesting. To handle these requests you can register custom display modes. We can do that using DisplayModeProvider.Instance.Modes.Insert(int index, IDisplayMode item) method.

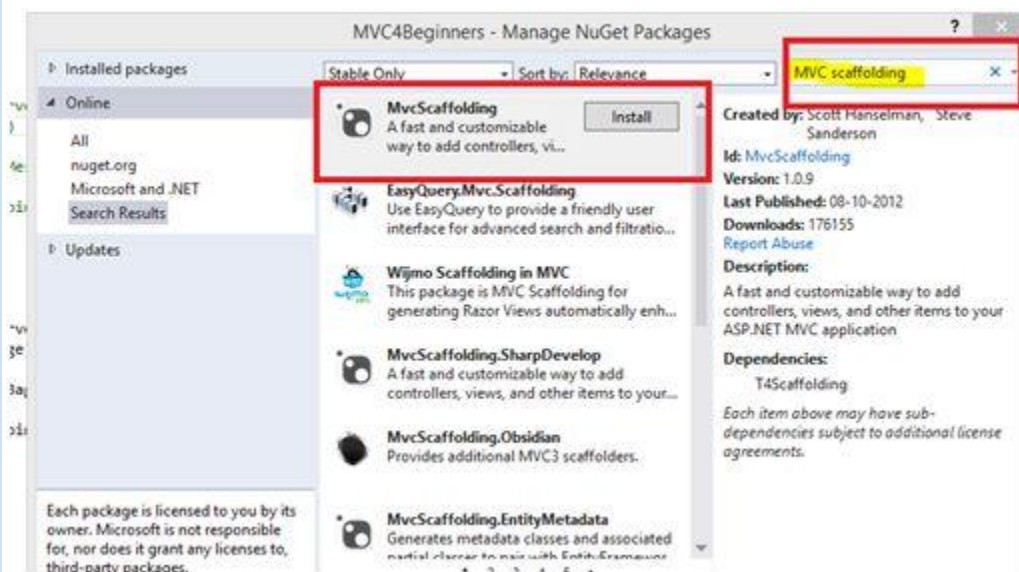
Learn more here - [Display Mode Provider in MVC 5 Application](#)

21. Explain the concept of MVC Scaffolding?

Answer

ASP.NET Scaffolding is a code generation framework for ASP.NET Web applications. Visual Studio 2013 includes pre-installed code generators for MVC and Web API projects. You add scaffolding to your project when you want to quickly add code that interacts with data models. Using scaffolding can reduce the amount of time to develop standard data operations in your project.

Scaffolding consists of page templates, entity page templates, field page templates, and filter templates. These templates are called Scaffold templates and allow you to quickly build a functional data-driven Website.



Scaffolding Templates



Create

It creates a View that helps in creating a new record for the Model. It automatically generates a label and input field for each property in the Model.

Delete

It creates a list of records from the model collection along with the delete link with delete record.

Details

It generates a view that displays the label and an input field of the each property of the Model in the MVC framework.

Edit

It creates a View with a form that helps in editing the current Model. It also generates a form with label and field for each property of the model.

List

It generally creates a View with the help of a HTML table that lists the Models from the Model Collection. It also generates a HTML table column for each property of the Model.

Learn more here - [Terminologies in MVC: Part 3 \(Scaffolding\)](#)

22. What is Route Constraints in MVC?

Answer

Routing is a great feature of MVC, it provides a REST based URL that is very easy to remember and improves page ranking in search engines.

This article is not an introduction to Routing in MVC, but we will learn a few features of routing and by implementing them we can develop a very flexible and user-friendly application. So, let's start without wasting valuable time.

Add constraint to URL

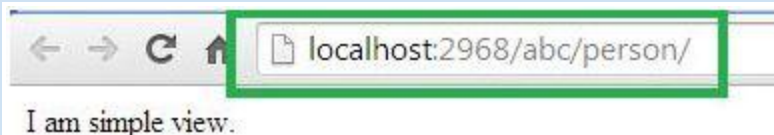
This is very necessary for when we want to add a specific constraint to our URL. Say, for example we want a [URL](#).

So, we want to set some constraint string after our host name. Fine, let's see how to implement it.

It's very simple to implement, just open the RouteConfig.cs file and you will find the routing definition in that. And modify the routing entry as in the following. We will see that we have added "abc" before.

```
17 routes.MapRoute(
18     name: "Default",
19     url: "abc/{controller}/{action}/{id}",
20     defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
21 );
22
23 }
```

Controller name, now when we browse we need to specify the string in the URL, as in the following:



Learn more here - [Route Constraints in MVC](#)

23. What is Razor View Engine in MVC?

Answer

ASP.NET MVC has always supported the concept of "view engines" that are the pluggable modules that implement various template syntax options. The "default" view engine for ASP.NET MVC uses the same .aspx/.ascx/.master file templates as ASP.NET Web Forms. In this article I go through the Razor View Engine to create a view of an application. "Razor" was in development beginning in June 2010 and was released for Microsoft Visual Studio in January 2011.

Razor is not a new programming language itself, but uses C# syntax for embedding code in a page without the ASP.NET delimiters: `<%= %>`. It is a simple-syntax view engine and was released as part of ASP.NET MVC 3. The Razor file extension is ".cshtml" for the C# language. It supports TDD (Test Driven Development) because it does not depend on the `System.Web.UI.Page` class.

Learn more here - [Getting Started with Razor View Engine in MVC 3](#)

24. What is Output Caching in MVC?

Answer

The main purpose of using Output Caching is to dramatically improve the performance of an ASP.NET MVC Application. It enables us to cache the content returned by any controller method so that the same content does not need to be generated each time the same controller method is invoked. Output Caching has huge advantages, such as it reduces server round trips, reduces database server round trips, reduces network traffic etc.

Keep the following in mind,

- Avoid caching contents that are unique per user.
- Avoid caching contents that are accessed rarely.
- Use caching for contents that are accessed frequently.

Let's take an example. My MVC application displays a list of database records on the view page so by default each time the user invokes the controller method to see records, the application loops through the entire process and executes the database query. And this can actually decrease the application performance. So, we can advantage of the "Output Caching" that avoids executing database queries each time the user invokes the controller method. Here the view page is retrieved from the cache instead of invoking the controller method and doing redundant work.

Cached Content Locations

In the above paragraph I said, in Output Caching the view page is retrieved from the cache, so where is the content cached/stored?

Please note, there is no guarantee that content will be cached for the amount of time that we specify. When memory resources become low, the cache starts evicting content automatically.

OutputCache label has a "Location" attribute and it is fully controllable. Its default value is "Any", however there are the [following locations](#) available; as of now, we can use any one.

1. Any
2. Client
3. Downstream
4. Server
5. None
6. ServerAndClient

With "Any", the output cache is stored on the server where the request was processed. The recommended store cache is always on the server very carefully. You will learn about some security related tips in the following "Don't use Output Cache".

Learn more here - [Output Caching in MVC](#)

25. What is Bundling and Minification in MVC?

Answer

Bundling and minification are two new techniques introduced to improve request load time. It improves load time by reducing the number of requests to the server and reducing the size of requested assets (such as CSS and JavaScript).

Bundling

It lets us combine multiple JavaScript (.js) files or multiple cascading style sheet (.css) files so that they can be downloaded as a unit, rather than making individual HTTP requests.

Minification

It squeezes out whitespace and performs other types of compression to make the downloaded files as small as possible. At runtime, the process identifies the user agent, for example IE, Mozilla, etc. and then removes whatever is specific to Mozilla when the request comes from IE.

Learn more here - [Bundling and Minification in Visual Studio 2012 or Migrate Existing Website](#)

26. What is Validation Summary in MVC?

Answer

The ValidationSummary helper method generates an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.

The `ValidationSummary` can be used to display all the error messages for all the fields. It can also be used to display custom error messages. The following figure shows how `ValidationSummary` displays the error messages.

Edit

Student

- The Name field is required.
- The Age field is required.

Name

Age

Save

ValidationSummary() Signature

MvcHtmlString ValidateMessage(bool excludePropertyErrors, string message, object htmlAttributes)

Display field level error messages using ValidationSummary

By default, `ValidationSummary` filters out field level error messages. If you want to display field level error messages as a summary then specify *excludePropertyErrors = false*.

Example - ValidationSummary to display field errors

```
@Html.ValidationSummary(false, "", new { @class = "text-danger" })
```

So now, the following Edit view will display error messages as a summary at the top. Please make sure that you don't have a `ValidationMessageFor` method for each of the fields.

Edit

Student

- The Name field is required.
- The Age field is required.

Name

Age

Save

Learn more here - [Understanding Validation In MVC - Part 4](#)

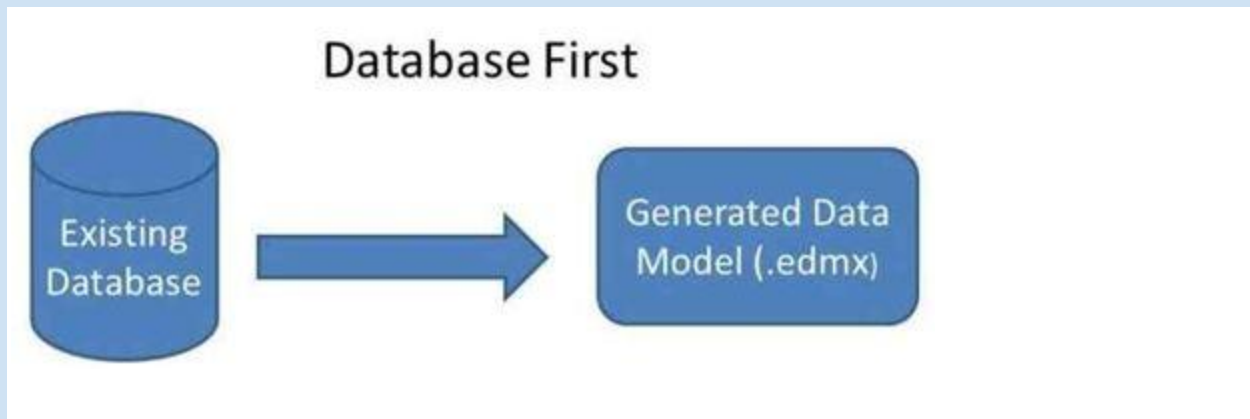
27. What is Database First Approach in MVC using Entity Framework?

Answer

Database First Approach is an alternative to the Code First and Model First approaches to the Entity Data Model which creates model codes (classes, properties, `DbContext` etc) from the database in the project and that classes behaves as the link between database and controller.

There are the following approach which is used to connect with database to application.

- Database First
- Model First
- Code First



Database first is nothing but only a approach to create web application where database is available first and can interact with the database. In this database, database is created first and after that we manage the code. The Entity Framework is able to generate a business model based on the tables and columns in a relational database.

Learn more here - [Database First Approach With ASP.NET MVC Step By Step Part 1](#)

28. What are the Folders in MVC application solutions?

Answer - Understanding the folders

When you create a project a folder structure gets created by default under the name of your project which can be seen in solution explorer. Below i will give you a brief explanation of what these folders are for.

Model

This folder contains classes that is used to provide data. These classes can contain data that is retrived from the database or data inserted in the form by the user to update the database.

Controllers

These are the classes which will perform the action invoked by the user. These classes contains methods known as "Actions" which responds to the user action accordingly.

Views

These are simple pages which uses the model class data to populate the HTML controls and renders it to the client browser.

App_Start

Contains Classes such as FilterConfig, RoutesConfig, WebApiConfig. As of now we need to understand the RouteConfig class. This class contains the default format of the url that should be supplied in the browser to navigate to a specified page.

29. What is difference between MVC and Web Forms?

Answer - ASP.Net MVC / Web Forms difference

The following are some difference.

ASP.Net MVC	ASP.Net Web Forms
View and logic are separate, it has separation of concerns theory. MVC 3 onwards has .aspx page as .cshtml.	No separation of concerns; Views are tightly coupled with logic (.aspx.cs /.vb file).
Introduced concept of routing for route based URL. Routing is declared in Global.asax for example.	File-based routing .Redirection is based on pages.
Support Razor syntax as well as .aspx	Support web forms syntax only.
State management handled via TempData, ViewBag, and View Data. Since the controller and view are not dependent and also since there is no view state concept in ASP.NET, MVC keeps the pages lightweight.	State management handled via View State. Large viewstate, in other words increase in page size.
Partial Views	User Controls
HTML Helpers	Server Controls
Multiple pages can have the same controller to satisfy their requirements. A controller may have multiple Actions (method name inside the controller class).	Each page has its own code, in other words direct dependency on code. For example Sachin.aspx is dependent on Sachin.aspx.cs (code behind) file.
Unit Testing is quite easier than ASP.Net Web forms Since a web form and code are separate files.	Direct dependency, tight coupling raises issues in testing.

Here are more [Similarities and Dissimilarities Between MVC and Web Forms](#).

30. What are the methods of handling an Error in MVC?

Answer

Exception handling may be required in any application, whether it is a web application or a Windows Forms application.

ASP.Net MVC has an attribute called "HandleError" that provides built-in exception filters. The HandleError attribute in ASP.NET MVC can be applied over the action method as well as Controller or at the global level. The HandleError attribute is the default implementation of `ExceptionHandler`. When we create a MVC application, the HandleError attribute is added within the Global.asax.cs file and registered in the Application_Start event.

```
public static void RegisterGlobalFilters(GlobalFilterCollection
filters) { filters.Add(new HandleErrorAttribute()); } protected void
Application_Start() { AreaRegistration.RegisterAllAreas();
RegisterGlobalFilters(GlobalFilters.Filters);
RegisterRoutes(RouteTable.Routes); }
```

C#Copy

Important properties of HandleError attribute

The HandleError attribute has a couple of properties that are very useful in handling the exception.

ExceptionType

Type of exception to be catch. If this property is not specified then the HandleError filter handles all exceptions.

View

Name of the view page for displaying the exception information.

Master

Master View for displaying the exception.

Order

Order in which the action filters are executed. The Order property has an integer value and it specifies the priority from 1 to any positive integer value. 1 means highest priority and the greater the value of the integer is, the lower is the priority of the filter.

AllowMultiple

It indicates whether more than one instance of the error filter attribute can be specified.

Example

```
[HandleError(View = "Error")] public class HomeController: Controller
{ public ActionResult Index() { ViewBag.Message = "Welcome to ASP.NET
MVC!"; int u = Convert.ToInt32(""); // Error line return View(); } }
```

C#Copy

HandleError Attribute at Action Method Level,

```
[HandleError(View = "Error")] public ActionResult Index() {
ViewBag.Message = "Welcome to ASP.NET MVC!"; int u =
Convert.ToInt32(""); // Error line return View(); }
```

C#Copy

Here are more details on [Exception or Error Handling in ASP.Net MVC Using HandleError Attribute](#).

31. How can we pass the data From Controller To View In MVC?

Answer

There are three options in Model View Controller (MVC) for passing data from controller to view. This article attempts to explain the differences among ViewData, ViewBag and TempData with examples. ViewData and ViewBag are similar and TempData performs additional responsibility. The following are the key points on those three objects.

ViewData

- The ViewData is used to move data from controller to view.
- The ViewData is a dictionary of objects that are derived from the "ViewDataDictionary" class and it will be accessible using strings as keys.
- ViewData contains a null value when redirection occurs.
- ViewData requires typecasting for complex data types.

ViewBag

- ViewBag is just a dynamic wrapper around ViewData and exists only in ASP.NET MVC 3. ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0.
- ViewBag doesn't require typecasting for complex data types.
- ViewBag also contain a null value when redirection occurs.

TempData

- ViewData moves data from controller to view.
- Use TempData when you need data to be available for the next request, only. In the next request, it will be there but will be gone after that.
- TempData is used to pass data from the current request to the subsequent request, in other words in case of redirection. That means the value of TempData will not be null.

Learn more here - [Various Ways to Pass Data From Controller to View in MVC 4](#)

32. What is Scaffolding in MVC?

Answer

Scaffolding is a code generation framework for ASP.NET Web applications. Visual Studio 2013 includes pre-installed code generators for MVC and Web API projects. You add scaffolding to your project when you want to quickly add code that interacts with data models. Using scaffolding can reduce the amount of time to develop standard data operations in your project.

Prerequisites

To use ASP.NET Scaffolding, you must have,

- Microsoft Visual Studio 2013
- Web Developer Tools (part of default Visual Studio 2013 installation)
- ASP.NET Web Frameworks and Tools 2013 (part of default Visual Studio 2013 installation)

What are the Advantages of using Scaffolding ?

- Minimal or no code to create a data-driven Web applications.
- Quick development time.
- Pages that are fully functional and include display, insert, edit, delete, sorting, and paging functionalities.
- Built-in data validation that is based on the database schema.
- Filters that are created for each foreign key or Boolean fields.

Learn more here - [Scaffolding In MVC 5](#)

33. What is ViewStart?

Answer

Razor View Engine introduced a new layout named `_ViewStart` which is applied on all view automatically. Razor View Engine firstly executes the `_ViewStart` and then start rendering the other view and merges them.

Example of Viewstart

```
@ { Layout = "~/Views/Shared/_v1.cshtml"; } < !DOCTYPE html > < html >  
< head > < meta name = "viewport" content = "width=device-width" / > <  
title > ViewStart < /title> < /head> < body > < /body> < /html>
```

MarkupCopy

Learn more here - [Viewstart Page in ASP.NET MVC 3](#)

34. What is JsonResultType in MVC?

Answer

Action methods on controllers return JsonResult (JavaScript Object Notation result) that can be used in an AJAX application. This class is inherited from the "ActionResult" abstract class. Here Json is provided one argument which must be serializable. The JSON result object that serializes the specified object to JSON format.

```
public JsonResult JsonResultTest() { return Json("Hello My Friend!");  
}
```

C#Copy

Learn more here - [ActionResult Return Type in MVC 3.0](#)

35. What is TempData?

Answer - TempData

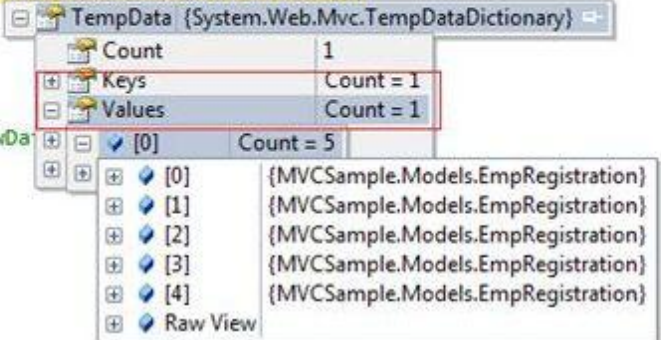
- TempData is a dictionary object derived from the TempDataDictionary class.
- TempData is used to pass data from the current request to a subsequent request, in other words in the case of redirection.
- The life of a TempData is very short and it retains its value for a short period of time.
- It requires typecasting for complex data type as I've used in my example:

- @foreach (var item in (List<MVCSample.Models.EmpRegistration>)TempData["EmployeeRegistration"])
- You can retain its value using the Keep method for subsequent requests.

```
public ActionResult Verify()
{
    TempData["EmployeeRegistration"] = ObjEmp.GetEmpRegistrationsDetails();
    TempData.Keep("EmployeeRegistration");
    return View("Verify", TempData["EmployeeRegistration"]);
}

#region Commented Code
//return View("Verify", ViewData);
#endregion Commented Code

//
// GET: /Register/Details/5
```



36. How to use ViewBag?

Answer

ViewBag is dynamic property that takes advantage of new dynamic features in C# 4.0. It's also used to pass data from a controller to a view. In short, The ViewBag property is simply a wrapper around the ViewData that exposes the ViewData dictionary as a dynamic object. Now create an action method "StudentSummary" in the "DisplayDataController" controller that stores a Student class object in ViewBag.

```
public ActionResult StudentSummary() { var student = new Student() {
    Name = "Sandeep Singh Shekhawat", Age = 24, City = "Jaipur" };
    ViewBag.Student = student; return View(); }
```

C#Copy

Thereafter create a view StudentSummary ("StudentSummary.cshtml") that shows student object data. ViewBag does not require typecasting for complex data type so you can directly access the data from ViewBag.

```
@ { ViewBag.Title = "Student Summary"; var student = ViewBag.Student;
} < table > < tr > < th > Name < /th> < th > Age < /th> < th > City <
/th> < /tr> < tr > < td > @student.Name < /td> < td > @student.Age <
/td> < td > @student.City < /td> < /tr> < /table>
```

C#Copy

Here we used one more thing, "ViewBag.Title", that shows the title of the page.

Learn more here - [Displaying Data on View From Controller](#)

37. What are the Difference between ViewBag&ViewData?

Answer - Difference between ViewBag&ViewData?

- ViewData is a dictionary of objects that is derived from ViewDataDictionary class and accessible using strings as keys.
- ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0.
- ViewData requires typecasting for complex data type and check for null values to avoid error.
- ViewBag doesn't require typecasting for complex data type.
- Calling of ViewBag is:
ViewBag.Name = "Yogesh";
Calling of ViewData is :
ViewData["Name"] = "yogesh";

Learn more here - [Difference Between ViewBag & ViewData in MVC](#)

38. What is Data Annotation Validator Attributes in MVC?

Answer - Using the Data Annotation Validator Attributes

DataAnnotation plays a vital role in added validation to properties while designing the model itself. This validation can be added for both the client side and the server side.

You understand that decorating the properties in a model with an Attribute can make that property eligible for Validation.

Some of the DataAnnotation used for validation are given below,

Error List:

- Minimum char is 5 and maximum char is 10
- CustomerName contains invalid character.
- Customer Code is too small
- Invalid character
- Range should be between 1k & 10k

Customer Code Customer Code is too small

Customer Name Minimum char is 5 and maximum char is 10

Amount Range should be between 1k & 10k

1. *Required*
Specify a property as required.
[Required(ErrorMessage="CustomerName is mandatory")]
2. *RegularExpression*
Specifies the regular expression to validate the value of the property.
[RegularExpression("[a-z]", ErrorMessage = "Invalid character")]
3. *Range*
Specifies the Range of values between which the property values are checked.
[Range(1000,10000,ErrorMessage="Range should be between 1k & 10k")]
4. *StringLength*
Specifies the Min & Max length for a string property.
[StringLength(50, MinimumLength = 5, ErrorMessage = "Minimum char is 5 and maximum char is 10")]
5. *MaxLength*
Specifies the Max length for the property value.
[MaxLength(10,ErrorMessage="Customer Code is exceeding")]
6. *MinLength*
It is used to check for minimum length.
[MinLength(5, ErrorMessage = "Customer Code is too small")]

Learn more here - [Adding Custom Validation in MVC](#)

39. How can we done Custom Error Page in MVC?

Answer

The `HandleErrorAttribute` allows you to use a custom page for this error. First you need to update your `web.config` file to allow your application to handle custom errors.

```
<system.web> <customErrors mode="On"> </system.web>
```

MarkupCopy

Then, your action method needs to be marked with the attribute.

```
[HandleError] public class HomeController: Controller { [HandleError]  
publicActionResultThrowException() { throw new ApplicationException();  
} }
```

C#Copy

By calling the `ThrowException` action, this would then redirect the user to the default error page. In our case though, we want to use a custom error page and redirect the user there instead. So, let's create our new custom view page.

```

<%@ Page Language="C#" Inherits="System.Web.Mvc.ViewPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>CustomErrorView</title>
</head>
<body>
    <h2>
        Error</h2>
    <p>
        Controller:
        <%= ((HandleErrorInfo) ViewData.Model) .ControllerName %>
    </p>
    <p>
        Action:
        <%= ((HandleErrorInfo) ViewData.Model) .ActionName %>
    </p>
    <p>
        Message:
        <%= ((HandleErrorInfo) ViewData.Model) .Exception.Message %>
    </p>
    <p>
        Stack Trace:
        <%= ((HandleErrorInfo) ViewData.Model) .Exception.StackTrace %>
    </p>
</body>
</html>

```

Next, we simply need to update the HandleErrorAttribute on the action method.

```

[HandleError] public class HomeController: Controller {
[HandleError(View = "CustomErrorView")]
public ActionResult ThrowException() { throw new ApplicationException();
} }

```

C#Copy

Learn more here - [Custom Error Page in ASP.NET MVC](#)

40. Server Side Validation in MVC?

Answer

The ASP.NET MVC Framework validates any data passed to the controller action that is executing. It populates a ModelState object with any validation failures that it finds and passes that object to the controller. Then the controller actions can query the ModelState to discover whether the request is valid and react accordingly.

I will use two approaches in this article to validate a model data. One is to manually add an error to the ModelState object and another uses the Data Annotation API to validate the model data.

Approach 1 - Manually Add Error to ModelState object

I create a User class under the Models folder. The User class has two properties "Name" and "Email". The "Name" field has required field validations while the "Email" field has Email validation. So let's see the procedure to implement the validation. Create the User Model as in the following,

```
namespace ServerValidation.Models { public class User { public string
Name { get; set; } public string Email { get; set; } } }
```

C#Copy

After that I create a controller action in User Controller (UserController.cs under Controllers folder). That action method has logic for the required validation for Name and Email validation on the Email field. I add an error message on ModelState with a key and that message will be shown on the view whenever the data is not to be validated in the model.

```
using System.Text.RegularExpressions; using System.Web.Mvc; namespace
ServerValidation.Controllers { public class UserController: Controller
{ public ActionResult Index() { return View(); } [HttpPost] public
ActionResult Index(ServerValidation.Models.User model) { if
(string.IsNullOrEmpty(model.Name)) { ModelState.AddModelError("Name",
"Name is required"); } if (!string.IsNullOrEmpty(model.Email)) {
string emailRegex = @ "^[a-zA-Z0-9_-\.\.]+)((\[[0-9]{1,3}" + @
```

```
"\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-\_]+\\" + @
".)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\")?)"$"; Regex re = new
Regex(emailRegex); if (!re.IsMatch(model.Email)) {
ModelState.AddModelError("Email", "Email is not valid"); } } else {
ModelState.AddModelError("Email", "Email is required"); } if
(ModelState.IsValid) { ViewBag.Name = model.Name; ViewBag.Email =
model.Email; } return View(model); } } }
```

C#Copy

Thereafter I create a view (Index.cshtml) for the user input under the User folder.

```
@model ServerValidation.Models.User @ { ViewBag.Title = "Index"; }
@using(Html.BeginForm()) { if (@ViewData.ModelState.IsValid) { if
(@ViewBag.Name != null) { < b > Name: @ViewBag.Name < br / > Email:
@ViewBag.Email < /b> } } < fieldset > < legend > User < /legend> < div
class = "editor-label" > @Html.LabelFor(model => model.Name) < /div> <
div class = "editor-field" > @Html.EditorFor(model => model.Name)
@if(!ViewData.ModelState.IsValid) { < span class =
"field-validation-error" >
@ViewData.ModelState["Name"].Errors[0].ErrorMessage < /span> } < /div>
< div class = "editor-label" > @Html.LabelFor(model => model.Email) <
/div> < div class = "editor-field" > @Html.EditorFor(model =>
model.Email) @if(!ViewData.ModelState.IsValid) { < span class =
"field-validation-error" >
@ViewData.ModelState["Email"].Errors[0].ErrorMessage < /span> } <
/div> < p > < input type = "submit" value = "Create" / > < /p> <
/fieldset> }
```

JavaScriptCopy

41. What is the use of remote validation in MVC?

Answer

Remote validation is the process where we validate specific data posting data to a server without posting the entire form data to the server. Let's see an actual scenario, in one of my projects I had a requirement to validate an email address, whether it already exists in the database. Remote validation was useful for that; without posting all the data we can validate only the email address supplied by the user.

Practical Explanation

Let's create a MVC project and name it accordingly, for me its "TestingRemoteValidation". Once the project is created let's create a model named UserModel that will look like:

```
public class UserModel { [Required] public string UserName { get; set; } [Remote("CheckExistingEmail", "Home", ErrorMessage = "Email already exists!")] public string UserEmailAddress { get; set; } }
```

C#Copy

Let's get some understanding of the remote attribute used, so the very first parameter "CheckExistingEmail" is the the name of the action. The second parameter "Home" is referred to as controller so to validate the input for the UserEmailAddress the "CheckExistingEmail" action of the "Home" controller is called and the third parameter is the error message. Let's implement the "CheckExistingEmail" action result in our home controller.

```
public ActionResult CheckExistingEmail(string UserEmailAddress)
{
    bool ifEmailExist = false;
    try
    {
        ifEmailExist = UserEmailAddress.Equals("mukeshknayak@gmail.com") ? true : false;
        return Json(!ifEmailExist, JsonRequestBehavior.AllowGet);
    } catch (Exception ex)
    {
        return Json(false, JsonRequestBehavior.AllowGet);
    }
}
```

}

Learn more here - [Remote Validation in MVC](#)

42. What are the Exception filters in MVC?

Answer

Exception are part and parcel of an application. They are a boon and a ban for an application too. Isn't it? This would be controversial, for developers it helps them track minor and major defects in an application and sometimes they are frustrating when it lets users land on the Yellow screen of death each time. This would make the users mundane to the application. Thus to avoid this, developers handle the exceptions. But still sometimes there are a few unhandled exceptions.

Now what is to be done for them? MVC provides us with built-in "Exception Filters" about which we will explain here.



cc: Google

Let's start!

A Yellow screen of Death can be said is as a wardrobe malfunction of our application.

Get Started

Exception filters run when some of the exceptions are unhandled and thrown from an invoked action. The reason for the exception can be anything and so is the source of the exception.

Creating an Exception Filter

Custom Exception Filters must implement the builtin `IExceptionFilter` interface. The interface looks as in the following,

```
public interface IExceptionFilter { void OnException(ExceptionContext filterContext) }
```

C#Copy

Whenever an unhandled exception is encountered, the `OnException` method gets invoked. The parameter as we can see, `ExceptionContext` is derived from the `ControllerContext` and has a number of built-in properties that can be used to get the information about the request causing the exception. Their property's `ExceptionContext` passess are shown in the following table:

Name	Type	Detail
Result	ActionResult	The result returned by the action being invoked.
Exception	Exception	The unhandled exceptions caused from the actions in the applications.
ExceptionHandled	BOOL	This is a very handy property that returns a bool value (true/false) based on if the exception is handled by any of the filters in the applicaiton or not.

The exception being thrown from the action is detailed by the `Exception` property and once handled (if), then the property `ExceptionHandled` can be toggled, so that the other filters would know if the exception has been already handled and cancel the other filter requests to handle. The problem is that if the exceptions are not handled, then the

default MVC behavior shows the dreaded yellow screen of death. To the users, that makes a very impression on the users and more importantly, it exposes the application's handy and secure information to the outside world that may have hackers and then the application gets into the road to hell. Thus, the exceptions need to be dealt with very carefully. Let's show one small custom exception filter. This filter can be stored inside the Filters folder in the web project of the solution. Let's add a file/class called CustomExceptionHandler.cs.

```
public class CustomExceptionHandler: FilterAttribute, IExceptionHandler
{
    public void OnException(ExceptionContext filterContext) {
        if (!filterContext.ExceptionHandled && filterContext.Exception is
        NullReferenceException) {
            filterContext.Result = new
            RedirectResult("customErrorPage.html");
            filterContext.ExceptionHandled
            = true;
        }
    }
}
```

C#Copy

Learn more here - [Exception Filters in MVC](#)

43. What is MVC HTML- Helpers and its Methods?

Answer

Helper methods are used to render HTML in the view. Helper methods generates HTML output that is part of the view. They provide an advantage over using the HTML elements since they can be reused across the views and also requires less coding.

There are several builtin helper methods that are used to generate the HTML for some commonly used HTML elements, like form, checkbox, dropdownlist etc. Also we can create our own helper methods to generate custom HTML. First we will see how to use the builtin helper methods and then we will see how to create custom helper methods.

Standard HtmlHelper methods

Some of the standard helper methods are,

- ActionLink: Renders an anchor.
- BeginForm: Renders HTML form tag
- CheckBox: Renders check box.
- DropDownList: Renders drop-down list.
- Hidden: Renders hidden field
- ListBox: Renders list box.
- Password: Renders TextBox for password input
- RadioButton: Renders radio button.
- TextArea: Renders text area.
- TextBox: Renders text box.

Learn more here - [HtmlHelper Methods in ASP.NET MVC](#)

44. Define Controller in MVC?

Answer

The controller provides model data to the view, and interprets user actions such as button clicks. The controller depends on the view and the model. In some cases, the controller and the view are the same object.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace DemoMVC.Controllers
{
    0 references
    public class EmployeeController : Controller
    {
        0 references
        public string EmployeeNames()
        {
            Action return @"<ul>
                <li>Sourabh Somani</li>
                <li>Shaili Dashora</li>
                <li>Saloni Choudhry</li>
                <li>Mahesh Chand</li>
                <li>DJ</li>
                <li>Dinesh Beniwal</li>
            </ul>";
        }
    }
}

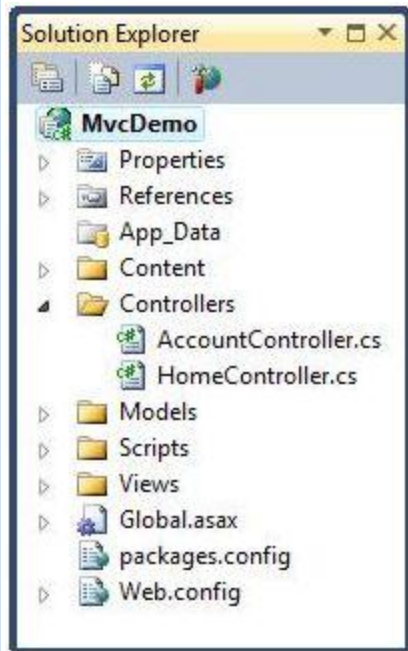
```

The Controllers Folder

The Controllers Folder contains the controller classes responsible for handling user input and responses. MVC requires the name of all controllers to end with "Controller".

In our example, Visual Web Developer has created the following files:

HomeController.cs (for the Home and About pages) and AccountController.cs (For the Log On pages):



Learn more here - [ASP.Net MVC Controller](#)

45. Explain Model in MVC?

Answer

The model represents the data, and does nothing else. The model does NOT depend on the controller or the view. The MVC Model contains all application logic (business logic, validation logic, and data access logic), except pure view and controller logic. With MVC, models both hold and manipulate application data.

The Models Folde

The Models Folder contains the classes that represent the application model.

Visual Web Developer automatically creates an AccountModels.cs file that contains the models for application security.

Learn more here - [Model in ASP.Net MVC : Part 1](#)

46. Explain View in MVC?

Answer

A view is responsible for displaying all of, or a portion of, data for users. In simple terms, whatever we see on the output screen is a view.

The Views Folder

The Views folder stores the files (HTML files) related to the display of the application (the user interfaces). These files may have the extensions html, asp, aspx, cshtml, and vbhtml, depending on the language content.

The Views folder contains one folder for each controller. Visual Web Developer has created an Account folder, a Home folder, and a Shared folder (inside the Views folder). The Account folder contains pages for registering and logging in to user accounts. The Home folder is used for storing application pages like the home page and the about page. The Shared folder is used to store views shared between controllers (master pages and layout pages).



Learn more here - [ASP.Net MVC View](#)

47. What is Attribute Routing in MVC?

Answer

A route attribute is defined on top of an action method. The following is the example of a Route Attribute in which routing is defined where the action method is defined.

In the following example, I am defining the route attribute on top of the action method

```
public class HomeController: Controller { //URL: /MvcTest
[Route("MvcTest")] public ActionResult Index() ViewBag.Message =
"Welcome to ASP.NET MVC!"; return View(); } }
```

C#Copy

Attribute Routing with Optional Parameter

We can also define an optional parameter in the URL pattern by defining a mark ("?",) to the route parameter. We can also define the default value by using parameter=value.

```
public class HomeController : Controller { // Optional URI Parameter
// URL: /MvcTest/ // URL: /MvcTest/0023654 [Route("MvcTest /{
customerName ?}")] public ActionResult OtherTest(string customerName){
ViewBag.Message = "Welcome to ASP.NET MVC!"; return View(); } //
Optional URI Parameter with default value // URL: /MvcTest/ // URL:
/MvcTest/0023654 [Route("MvcTest /{ customerName =0036952}")] public
ActionResult OtherTest(string customerName) { ViewBag.Message =
"Welcome to ASP.NET MVC!"; return View(); } }
```

C#Copy

Learn more here - [Attribute Routing in ASP.Net MVC 5.0](#)

48. Explain RenderSection in MVC?

Answer

RenderSection() is a method of the WebPageBase class. Scott wrote at one point, The first parameter to the "RenderSection()" helper method specifies the name of the section we want to render at that location in the layout template. The second parameter is optional, and allows us to define whether the section we are rendering is required or not. If a section is "required", then Razor will throw an error at runtime if that section is not implemented within a view template that is based on the layout file (that can make it easier to track down content errors). It returns the HTML content to render.

```
<div id="body"> @RenderSection("featured", required: false) <section
class="content-wrapper main-content clear-fix"> @RenderBody()
</section> </div>
```

MarkupCopy

Learn more here - [ASP.Net MVC 4 - Layout and Section in Razor](#)

49. What is GET and POST Actions Types?

Answer

GET

GET is used to request data from a specified resource. With all the GET request we pass the URL which is compulsory, however it can take the following overloads.

```
.get(url [, data ] [, success(data, textStatus, jqXHR) ] [, dataType ] ).done/.fail
```

POST

POST is used to submit data to be processed to a specified resource. With all the POST requests we pass the URL which is compulsory and the data, however it can take the following overloads.

```
.post(url [, data ] [, success(data, textStatus, jqXHR) ] [, dataType ] )
```

Learn more here - [GET and POST Calls to Controller's Method in MVC](#)

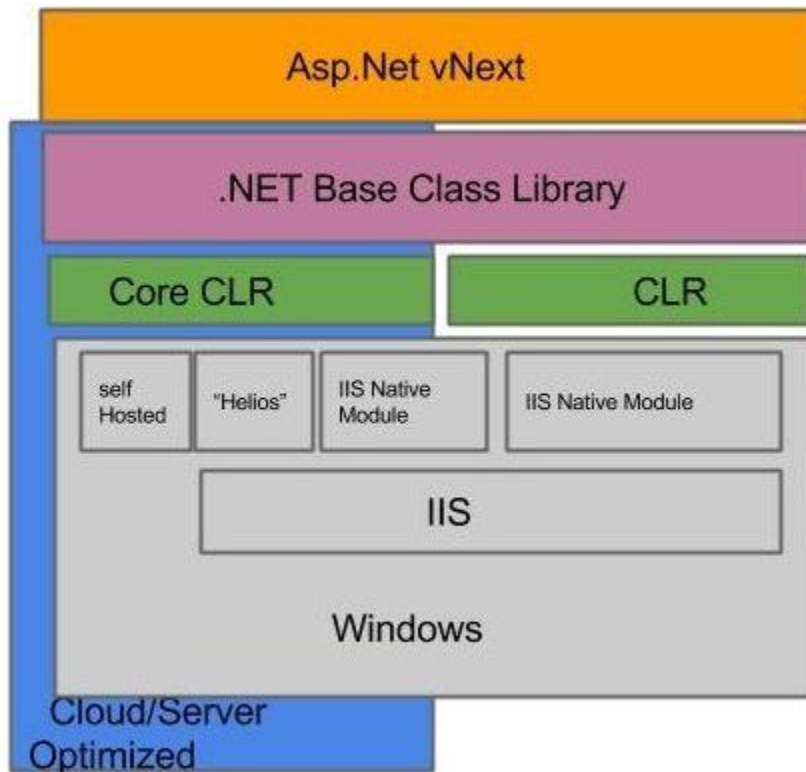
50. What's new in MVC 6?

Answer

In MVC 6 Microsoft removed the dependency of System.Web.Dll from MVC6 because it's so expensive that typically it consume 30k of memory per request and response, whereas now MVC 6 only requires 2k of memory per request and the response is a very small memory consumption.

The advantage of using the cloud-optimized framework is that we can include a copy of the mono CLR with your website. For the sake of one website we do not need to upgrade the .NET version on the entire machine. A different version of the CLR for a different website running side by side.

MVC 6 is a part of ASP.NET 5 that has been designed for cloud-optimized applications. The runtime automatically picks the correct version of the library when our MVC application is deployed to the cloud.



The Core CLR is also supposed to be tuned with a high resource-efficient optimization.

Microsoft has made many MVC, Web API, WebPage and SignalR pieces we call MVC 6.

Most of the problems are solved using the Roslyn Compiler. In ASP.NET vNext uses the Roslyn Compiler. Using the Roslyn Compiler we do not need to compile the application, it automatically compiles the application code. You will edit a code file and can then see the changes by refreshing the browser without stopping or rebuilding the project.

Run on hosts other than IIS

Where we use MVC5 we can host it on an IIS server and we can also run it on top of an ASP. NET Pipeline, on the other hand MVC 6 has a feature that makes it better and that feature is itself hosted on an IIS server and a self-user pipeline.

Environment based configuration system

The configuration system provides an environment to easily deploy the application on the cloud. Our application works just like a configuration provider. It helps to retrieve the value from the various configuration sources like XML file.

MVC 6 includes a new environment-based configuration system. Unlike something else it depends on just the Web.Config file in the previous version.

Dependency injection

Using the IServiceProvider interface we can easily add our own dependency injection container. We can replace the default implementation with our own container.

Supports OWIN

We have complete control over the composable pipeline in MVC 6 applications. MVC 6 supports the OWIN abstraction.