1) a)

Configure apps by using configuration files:-

In this article

.NET Framework gives developers and administrators control and flexibility over the way applications run through *configuration files*. Configuration files are XML files that can be changed as needed. An administrator can control which protected resources an application can access, which versions of assemblies an application will use, and where remote applications and objects are located. Developers can put settings in configuration files, eliminating the need to recompile an application every time a setting changes. This section describes what can be configured and why configuring an application might be useful.

Note

Managed code can use the classes in the **System.Configuration** namespace to read settings from the configuration files, but not to write settings to those files.

This article describes the syntax of configuration files and provides information about the three types of configuration files: machine, application, and security.

Configuration file format

Configuration files contain elements, which are logical data structures that set configuration information. Within a configuration file, you use tags to mark the beginning and end of an element. For example, the `<runtime>` element consists of `<runtime>`*child elements*`</runtime>`. An empty element would be written as `<runtime/>` or `<runtime></runtime>`.

As with all XML files, the syntax in configuration files is case-sensitive.

You specify configuration settings using predefined attributes, which are name/value pairs inside an element's start tag. The following example specifies two attributes

(`version` and `href`) for the `<codeBase>` element, which specifies where the runtime can locate an assembly (for more information, see Specifying an Assembly's Location).

XMLCopy

```
<codeBase version="2.0.0.0"
          href="http://www.litwareinc.com/myAssembly.dll"/>
```

Machine configuration files

The machine configuration file, Machine.config, contains settings that apply to an entire computer. This file is located in the *%runtime install path*%\Config directory. Machine.config contains configuration settings for machine-wide assembly binding, built-in remoting channels, and ASP.NET.

The configuration system first looks in the machine configuration file for the **<appSettings>** element and other configuration sections that a developer might define. It then looks in the application configuration file. To keep the machine configuration file manageable, it is best to put these settings in the application configuration file. However, putting the settings in the machine configuration file can make your system more maintainable. For example, if you have a third-party component that both your client and server application uses, it is easier to put the settings for that component in one place. In this case, the machine configuration file is the appropriate place for the settings, so you don't have the same settings in two different files.

Note

Deploying an application using XCOPY will not copy the settings in the machine configuration file.

For more information about how the common language runtime uses the machine configuration file for assembly binding, see How the Runtime Locates Assemblies.

Application configuration files

An application configuration file contains settings that are specific to an app. This file includes configuration settings that the common language runtime reads (such as assembly binding policy, remoting objects, and so on), and settings that the app can read.

The name and location of the application configuration file depend on the app's host, which can be one of the following:

- Executable–hosted app.

  These apps have two configuration files: a source configuration file, which is modified by the developer during development, and an output file that is distributed with the app.

  When you develop in Visual Studio, place the source configuration file for your app in the project directory and set its **Copy To Output Directory** property to **Copy always** or **Copy if newer**. By default, the name of the configuration file is *App.config*.

  To create the output configuration file that's deployed with the app, Visual Studio copies the source configuration file to the directory where the compiled assembly is placed. This file is named *<yourappname>.exe.config*. For example, an app named *myApp.exe* will have an output configuration file named *myApp.exe.config*.

  In some cases, Visual Studio may modify the output configuration file; for more information, see the Redirecting assembly versions at the app level section of the Redirecting Assembly Versions article.

- ASP.NET-hosted app.

  For more information about ASP.NET configuration files, see ASP.NET Configuration Settings.

- Internet Explorer-hosted app.

  If an app hosted in Internet Explorer has a configuration file, the location of this file is specified in a `<link>` tag with the following syntax: `<link rel="*ConfigurationFileName*" href="*location*">`

  In this tag, `location` is a URL to the configuration file. This sets the app base. The configuration file must be located on the same website as the app.

Security configuration files

Security configuration files contain information about the code group hierarchy and permission sets associated with a policy level. We strongly recommend that you use the Code Access Security Policy tool (Caspol.exe) to modify security policy to ensure that policy changes do not corrupt the security configuration files.

1) b)

The World Wide Web, or 'the web', in short, is an Internet-based global information system It makes available multimedia information from over 4 million computers around the world. the web offers video, interactive multimedia, and live audio, in addition to more basic data types, such as text documents and photographs.

the Internet.

Evolution of World Wide Web

World Wide Web (WWW) is a huge collection of hypertext pages on the Internet. The concept of WWW was developed in Switzerland at the European Particle Research Centre (Known as CERN), in the year 1989. The first text-based prototype was operational in 1991. in the month of December 1991, a public demonstration was given at Hypertext 91 conference in San Antonio, Texas (USA). In the year 1993, the first graphical interface software package called Mosaic was released.

Hypertext enables you to read and navigate the text and visual information in a nonlinear way based on what you want to know next, unlike a textbook where the subject is described continuously or linearly.

The Mosaic became so popular that a year later, the author of Mosaic namely, Marc Andressen left the National Center for Supercomputing Applications, where Mosaic was developed forming a company called Netscape Communications Corporation. This company developed the clients, servers, and other Web software.

In the year 1994, CERN and MIT of USA signed an agreement setting up the World Wide Web Consortium, an organization devoted to further developing the Web, standardizing protocols, and interoperability between sites. since this time, hundreds of universities and companies have joined the Consortium. In the first year after Mosaic was released the number of WWW servers grew from 100 to 7000. The growth is expected to be exponential in the years to come and will probably be the force driving the technology and use of the Internet into every walk of life of human beings.

the Web servers on the Internet are collectively referred to as the World Wide Web. The @3 Consortium is the closest anyone gets to setting the standards for and enforcing rules about the All Worldwide Web. you can visit the Consortium's home page at http:/www.w3.org/. the second group of organizations that influences the Web is the browser developers themselves, most notably NNetscape communications Corporation and Microsoft Corporation of USA.

To access the Web server, we use client software called a browser program. with a browser, we can choose an element on the Web page, which can then cross-link us to computer animation, or play sound, or show another Web page. The browser can even contact another Web server located across the world.

1) C)

Differences between Verification and Validation

Prerequisite – Verification and Validation

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing.
Verification means **Are we building the product right?**

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.
Validation means **Are we building the right product?**

The difference between Verification and Validation is as follow:

Verification	Validation
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.
It does *not* include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.

Verification	Validation
It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer

1) d)

Difference between ASP and ASP.NET

ASP: ASP stands for **Active Server Pages**. It is a development framework used for building web pages. ASP was introduced in 1998 by Microsoft as its first server-side scripting language. The file extension of ASP pages are .asp and are normally written in VBScript. It is an old but still powerful tool for making dynamic web pages. ASP is a technology (much like PHP) for executing scripts on a web server.
Example:

html

```
<html>

<body>
    <%response.write("Welcome to GeeksforGeeks!")%>
</body>

</html>
```

Output:

```
Welcome to GeeksforGeeks!
```

ASP.NET: ASP.NET was released in 2002 by Microsoft as a successor to ASP. It is also a server-side web framework, open-source, which is designed for the generation of dynamic web pages. The file extension of ASP.NET pages is .aspx and is normally written in C# (C sharp). The latest version of ASP.NET is ASP.NET 4.6.

Example:

csharp

```
    var rank = 50;
}
<html>
<body>
@if (rank < 60)
{
    <p>Welcome to GeeksforGeeks!</p>
}
</body>
</html>
```

Output:

```
Welcome to GeeksforGeeks!
```

Difference between ASP and ASP.NET:

ASP	ASP.NET
ASP is the interpreted language.	ASP.NET is the compiled language.
ASP uses **ADO** (ActiveX Data Objects) technology to connect and work with databases.	ASP.NET uses **ADO.NET** to connect and work with databases.
ASP is partially object-oriented.	ASP.NET is fully object-oriented.
In ASP there is no facility to separate design from programming logic.	In ASP.NET it has the option of Code Containment.

ASP	ASP.NET
ASP Pages have the file extension **.asp**.	ASP.NET Pages have the file extension **.aspx**.
ASP doesn't have the concept of inheritance.	ASP.NET inherit the class written in code behind.
ASP pages use scripting language.	ASP.NET uses full-fledged programming language.
Error handling is very poor in ASP.	Error handling is very good in ASP.NET.
In ASP debugging is difficult because the ASP scripts are interpreted.	In ASP.NET debugging is easy.
ASP is not a configurable language.	In ASP.NET Web.config is used for configuration.
ASP has maximum four in-built classes i.e. Request, Response, Session and Application.	ASP.NET has more than 2000 in-built classes.

1) e)

ASP.NET Page Life Cycle Events
At each stage of the page life cycle, the page raises some events, which could be coded. An event handler is basically a function or subroutine, bound to the event, using declarative attributes such as Onclick or handle.
Following are the page life cycle events:
PreInit - PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls, and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page_PreInit handler.
Init - Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page_Init handler.
InitComplete - InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
LoadViewState - LoadViewState event allows loading view state information into the controls.
LoadPostData - During this phase, the contents of all the input fields are defined with the <form> tag are processed.
PreLoad - PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page_PreLoad handler.

Load - The Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page_Load handler.
LoadComplete - The loading process is completed, control event handlers are run, and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page_LoadComplete handler
PreRender - The PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.
PreRenderComplete - As the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.
SaveStateComplete - State of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page_Render handler.
• UnLoad - The UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page_UnLoad handeler.

1) F)

ASP.NET HTML Server Controls

HTML server controls are HTML elements that contain attributes to accessible at server side. By default, HTML elements on an ASP.NET Web page are not available to the server. These components are treated as simple text and pass through to the browser. We can convert an HTML element to server control by adding a **runat="server"** and an **id** attribute to the component.

Now, we can easily access it at code behind.

Example

```
1. <input id="UserName" type="text" size="50"runat="server" />
```

All the HTML Server controls can be accessed through the **Request** object.

HTML Components

The following table contains commonly used HTML components.

Controls Name	Description
Button	It is used to create HTML button.
Reset Button	It is used to reset all HTML form elements.
Submit Button	It is used to submit form data to the server.
Text Field	It is used to create text input.
Text Area	It is used to create a text area in the html form.
File	It is used to create a input type = "file" component which is used to upload file to the server.
Password	It is a password field which is used to get password from the user.
CheckBox	It creates a check box that user can select or clear.
Radio Button	A radio field which is used to get user choice.
Table	It allows us to present information in a tabular format.
Image	It displays an image on an HTML form
ListBox	It displays a list of items to the user. You can set the size from two or more to specify how many items you wish to show.

Dropdown	It displays a list of items to the user in a dropdown list.
Horizontal Rule	It displays a horizontal line across the HTML page.

1) g)

Net Framework is a platform that provides tools and technologies to develop Windows, Web and Enterprise applications. It mainly contains two components,

1. Common Language Runtime (CLR)
2. .Net Framework Class Library.

1. Common Language Runtime (CLR)

.Net Framework provides runtime environment called Common Language Runtime (CLR).It provides an environment to run all the .Net Programs. The code which runs under the CLR is called as Managed Code. Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides memory management and thread management.

Programmatically, when our program needs memory, CLR allocates the memory for scope and de-allocates the memory if the scope is completed.

Language Compilers (e.g. C#, VB.Net, J#) will convert the Code/Program to Microsoft Intermediate Language (MSIL) intern this will be converted to Native Code by CLR. See the below Fig.

There are currently over 15 language compilers being built by Microsoft and other companies also producing the code that will execute under CLR.

2. .Net Framework Class Library (FCL)

This is also called as Base Class Library and it is common for all types of applications i.e. the way you access the Library Classes and Methods in VB.NET will be the same in C#, and it is common for all other languages in .NET.

The following are different types of applications that can make use of .net class library.

1. Windows Application.
2. Console Application
3. Web Application.
4. XML Web Services.
5. Windows Services.

In short, developers just need to import the BCL in their language code and use its predefined methods and properties to implement common and complex functions like reading and writing to file, graphic rendering, database interaction, and XML document manipulation.

Below are the few more concepts that we need to know and understand as part of this .Net framework.

3. Common Type System (CTS)

It describes set of data types that can be used in different .Net languages in common. (i.e), CTS ensures that objects written in different .Net languages can interact with each other.

For Communicating between programs written in any .NET complaint language, the types have to be compatible on the basic level.

The common type system supports two general categories of types:

Value types:

Value types directly contain their data, and instances of value types are either allocated on the stack or allocated inline in a structure. Value types can be built-in (implemented by the runtime), user-defined, or enumerations.

Reference types:

Reference types store a reference to the value's memory address, and are allocated on the heap. Reference types can be self-describing types, pointer types, or interface types. The type of a reference type can be determined from values of self-describing types. Self-describing types are further split into arrays and class types. The class types are user-defined classes, boxed value types, and delegates.

4. Common Language Specification (CLS)

It is a sub set of CTS and it specifies a set of rules that needs to be adhered or satisfied by all language compilers targeting CLR. It helps in cross language inheritance and cross language debugging.

Common language specification Rules:

It describes the minimal and complete set of features to produce code that can be hosted by CLR. It ensures that products of compilers will work properly in .NET environment.

Sample Rules:

1. Representation of text strings
2. Internal representation of enumerations
3. Definition of static members and this is a subset of the CTS which all .NET languages are expected to support.
4. Microsoft has defined CLS which are nothing but guidelines that language to follow so that it can communicate with other .NET languages in a seamless manner

1) h)

A web browser (also referred to as an Internet browser or simply a browser) is application software for accessing the World Wide Web or a local website. When a user requests a web page from a particular website, the web browser retrieves the necessary content from a web server and then displays the page on the user's device.

Features of a Web Browser

Some of the features of the web browser include-

Home button – Clicking the ‘Home’ button brings the user directly back to the home page of the browser. We can set any webpage as the home page. Usually people prefer to have search engines like – Google.com as their home page.

Address bar – The address bar is where the URL of the desired website is entered. This bar helps us to navigate to the desired website of our choice.

Refresh button – The refresh button is to reload the page. In some cases, the page locally stores and saves the information. This prevents users from seeing the updated information. Therefore, the refresh button is helpful in such cases.

Bookmarks – This option is to save a particular website for reference later in the future. It is used to mark pages that might be important or prove to be useful in the future.

Tabbed browsing – This feature helps to open new screens on the same browser for multiple browsing at the same time.

1) a)
   Different types of Configuration files

Machine.config - Server or machine-wide configuration file

Web.config - Application configuration files which deal with a single application

Machine.config File

Configuration files are applied to an executing site based on a hierarchy. There is a global configuration file for all sites in a given machine which is called Machine.config. This file is typically found in the C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\CONFIG directory.

The Machine.config file contains settings for all sites running on the machine provided another .config further up the chain does not override any of these settings. Although Machine.config provides a global configuration option, you can use .config files inside individual website directories to provide more granular control. Between these two poles you can set a number of other .config files with varying degree of applicable scope.

Application Configuration file (Web.config)

Each and Every ASP.NET application has its own copy of configuration settings stored in a file called Web.config. If the web application spans multiple folders, each sub folder has its own Web.config file that inherits or overrides the parent's file settings.

Processing of a Web.config file

When you initially run your web application, the runtime builds a cache of the configuration settings for your web application by flattening the layer of configuration files as below,

The Machine.config file settings are retrieved.

The settings from the root Web.config files are added to the caches, overwriting any conflicting settings that were earlier while reading the Machine.config file.

If there is a Web.config file at the root of the website, this file is read into the cache, all overwriting any existing settings. The resulting cache contains the setting for this website.

If you have subdirectories in your web application, each subdirectory can have a Web.config file that includes settings that are specific to the files and folders that are contained within the subdirectory. To calculate the effective setting for the folders, the website settings are read (step 1-4) and then this Web.config file is read into cache for this folder, overwriting (and thereby overriding) any existing settings.