**What are page directives?**

The first line of an ASP.NET page is the page directive; you will find it on all ASP.NET pages. These directives are instructions for the page. It begins with the @Page directive and continues with the various attributes available to this directive.

It's unreasonable to expect a candidate to know all of these attributes, but a few popular ones include the following.

* **AutoEventWireup:** Indicates whether page events are autowired.
* **CodeBehind:** The name of the compiled class associated with the page.
* **Debug:** Indicates whether the page is compiled in debug mode (includes debug symbols).
* **EnableTheming:** Indicates whether themes are used on the page.
* **EnableViewState:** Indicates whether view state is maintained across pages.
* **ErrorPage:** Specifies a target URL to be used when unhandled exceptions occur.
* **Language:** Indicates the language used when compiling inline code on the page.
* **Trace:** Signals whether tracing is enabled on the page.

## ****What is a master page?****

A master page is a template for one or more Web Forms. The master page defines how the page will be laid out when presented to the user, with placeholders for content. The MasterPageFile Page Directive in a content page's header is one way to assign a master page. The content pages rely solely on content and leave layout to the master page. ASP.NET merges the content with the master page layout when the content page is requested by a user.

## ****What is the code behind feature of ASP.NET?****

The code behind feature divides ASP.NET page files into two files where one defines the user interface (.aspx), while the other contains all of the logic or code (.aspx.cs for C# and .aspx.vb for VB.NET). These two files are glued together with page directives like the following line, which ties the page to the specific code behind class.

<%@ Page language="c#" Codebehind="UICode.cs" Inherits="Library.UICode" %>

## ****What are ASP.NET Server controls?****

ASP.NET includes a number of built-in server controls that are the foundation of its Web programming model. They have various properties to control their behavior and appearance. These controls provide an event model where events are handled on the server (whereas HTML controls are handled in the client). Server controls have the ability to maintain state (via ViewState) across requests, and they can automatically detect the browser. With these controls, you will see the RunAt attribute (RunAt="Server") that signals its processing will be done on the server. A good example of these controls is the basic TextBox control (<ASP:TextBox RunAt="Server" .... >.

## ****How can you loop through all controls on an ASP.NET Web form?****

You can easily traverse all controls on a form via the Web Form's Controls collection. The GetType method can be used on each control to determine its type and how to work with it. Now, it gets tricky because the form contains a tree of controls; that is, some controls are contained within others (think of a Table). You would have to recursively loop through the controls to make sure everything is processed.

## ****What is a web.config file? Machine.config?****

The web.config is the basic configuration file for ASP.NET applications. It utilizes an XML format. It is used to define application settings, connection strings, and much more. These files can appear in multiple directories, and they are applied in a top-down approach; that is, configuration files apply to their container directory as well as all directories below it, but the configuration files in lower directories can override those in parent directories. This provides a way to granularly apply settings. The machine.config file contains ASP.NET settings for all of the applications on the server -- it is at the top of the configuration file hierarchy, thus web.configs can override it.

**What is the use of @ Register directives?**  
  
@Register directive informs the compiler of any custom server control added to the page.  
  
**What is the use of Smart Navigation property?**  
  
It’s a feature provided by ASP.NET to prevent flickering and redrawing when the page is posted back.  
  
**What is AppSetting Section in “Web.Config” file?**  
  
Web.config file defines configuration for a web project. Using “AppSetting” section, we can define user-defined values. Example below is a “Connection String” section, which will be used throughout the project for database connection.\  
  
<Configuration>  
<appSettings>  
<add key="ConnectionString" value="server=xyz;pwd=www;database=testing" />  
</appSettings>  
  
**Where is View State information stored?**  
  
In HTML Hidden Fields.

**How can we create custom controls in ASP.NET?**  
  
User controls are created using .ASCX in ASP.NET. After .ASCX file is created you need to two things in order that the ASCX can be used in project:.

* Register the ASCX control in page using the <percentage@ Register directive.Example

<%@ Register tag prefix="Accounting" Tag name="footer" Src="Footer.ascx" %>

* Now to use the above accounting footer in page you can use the below directive.

<Accounting: footer runat="server" />

**What is inline function**

Using this syntax, you can use inline code in ASP.NET (.aspx) pages. The server-side code will be automatically compiled by the .NET framework the first time the page is requested on the server. The compiled .dll file is stored in the "Temporary ASP.NET Files" system folder. Changing the code in .aspx files will trigger a new compilation, generating new .dll files. The old .dll files are phased out by the framework and eventually deleted.  
  
**C#**

<%@ Import Namespace="System" %>  
<%@ Page Language="c#"%>  
  
<script runat="server">  
  public string ServerSideFunction(string input)  
  {  
    return "Hello " + input;  
  }  
</script>  
  
<% string pageVariable = "world"; %>  
<!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" xml:lang="en" lang="en">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252" />  
<title>ASP.NET inline</title>  
</head>  
<body>  
<% =ServerSideFunction(pageVariable) %>  
</body>  
</html>

**What is inline code?**

**Inline Code**

Inline Code refers to the code that is written inside an ASP.NET Web Page that has an extension of .aspx. It allows the code to be written along with the HTML source code using a <Script> tag. It's major point is that since it's physically in the .aspx file it's deployed with the Web Form page whenever the Web Page is deployed.

**What is Namespace?**

* Namespace is logical division of class, structure and interface OR way to organize your [Visual Basic](javascript:void(0);) .NET code is through the use of namespaces.
* Namespaces are a way of grouping type names and reducing the chance of name collisions.
* The namespace with all the built-in functionality comes under **System** namespace. All other namespaces comes under this **System** namespace.

# Declaration of Namespaces:

// Namespace Declaration in C#

* using System;
* using System.Data;

// Namespace Declaration in VB

* imports system;
* imports system.Data;

# Code Example :

namespace ExampleNamespace { class TestExample { public void ShowMessage() { Console.WriteLine("This is the TestExample namespace!"); } } }

# Advantages:

* We can establish security, version, reference, and deployment boundaries by using namespaces
* Because of grouping of namespaces we can create hierarchy which is easy to identify classes by fully qualified names.

**What is Globalization, Localization** ???

**Globalization, Localization**

To implementing a multilingual user interface, you design the user interface to open in the default UI language and offer the option to change to other languages.

Globalization is the first step in the process. A globalized application supports localized user interfaces and regional data for all users. Truly global applications should be culture-neutral and language-neutral. A globalized application can correctly accept, process, and display a worldwide assortment of scripts, data formats, and languages.

While your globalized application may possess such flexibility, ensure that you have separated the application's resources that require translation from the rest of the application's code. If you correctly test for localizability before proceeding to the localization step, you should not need to modify your application's source code during localization.

**Globalization in .Net: CultureInfo, ResourceManager and Resgen**

In .Net the System.Globalization namespace contains classes that define culture-related information, including the language, the country/region, the calendars in use, the format patterns for dates, currency and numbers and the sort order for strings. There are around 20 classes in System.Globalization. Here we are only concern to CutureInfo class.

The CutureInfo class represents information about a specific culture including the names of the culture, the writing system, and the calendar used, as well as access to culture-specific objects that provide information for common operations, such as formatting dates and sorting strings.

The System.Resources namespace provides classes and interfaces that allow developers to create, store, and manage various culture-specific resources used in an application. One of the most important classes of the System.Resources namespace is the ResourceManager class. The ResourceManager class allows the user to access and control resources stored in the main assembly or in resource satellite assemblies

The Resource File Generator (Resgen.exe) converts .txt files and .resx (XML-based resource format) files to common language runtime binary .resources files that can be embedded in a runtime binary executable or compiled into satellite assemblies.

**What is Global.asax?**

The Global.asax file, also known as the ASP.NET application file, is an optional file that contains code for responding to application-level events raised by ASP.NET or by HttpModules.

The Global.asax file resides in the root directory of an ASP.NET-based application. The Global.asax file is parsed and dynamically compiled by ASP.NET.

The Global.asax file itself is configured so that any direct URL request for it is automatically rejected; external users cannot download or view the code written within it.

The Global.asax file does not need recompilation if no changes have been made to it. There can be only one Global.asax file per application and it should be located in the application's root directory only.

The Global.asax contains two types of events those are

**Events which are fired for every request**

**Events which are not fired for every request**

Now I will explain about

**Events which are fired for every request**

**Application\_BeginRequest()** – This event raised at the start of every request for the web application.

**Application\_AuthenticateRequest** – This event rose just before the user credentials are authenticated. We can specify our own authentication logic here to provide custom authentication.

**Application\_AuthorizeRequest()** – This event raised after successful completion of authentication with user’s credentials. This event is used to determine user permissions. You can use this method to give authorization rights to user.

**Application\_ResolveRequestCache()** – This event raised after completion of an authorization request and this event used in conjunction with output caching. With output caching, the rendered HTML of a page is reused without executing its code.

**Application\_AcquireRequestState()** – This event raised just before session-specific data is retrieved for the client and is used to populate Session Collection for current request.

**Application\_PreRequestHandlerExecute()** – This event called before the appropriate HTTP handler executes the request.

**Application\_PostRequestHandlerExecute()** – This event called just after the request is handled by its appropriate HTTP handler.

**Application\_ReleaseRequestState()** – This event raised when session specific information is about to serialized from the session collection.

**Application\_UpdateRequestCache()** – This event raised just before information is added to output cache of the page.

**Application\_EndRequest()** – This event raised at the end of each request right before the objects released.

Now we will see

**Events which are not fired for every request**

**Application\_Start()** – This event raised when the application starts up and application domain is created.

**Session\_Start()** – This event raised for each time a new session begins, This is a good place to put code that is session-specific.

**Application\_Error()** – This event raised whenever an unhandled exception occurs in the application. This provides an opportunity to implement generic application-wide error handling.

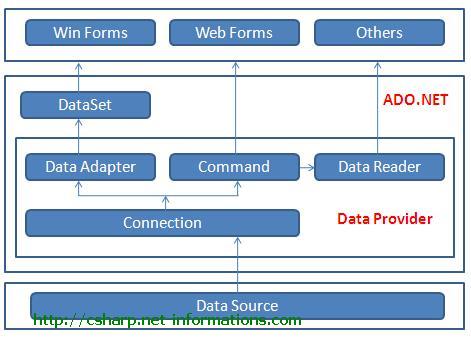
**Session\_End()** – This event called when session of user ends.

**Application\_End()** – This event raised just before when web application ends.

**Application\_Disposed()** – This event fired after the web application is destroyed and this event is used to reclaim the memory it occupies.

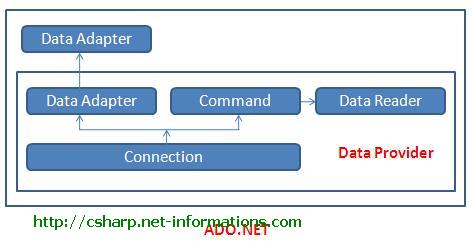
**What is ado.net**

**ADO.NET**



**ADO.NET** is a data access technology from Microsoft [.Net Framework](http://vb.net-informations.com/framework/what_is_net_framework.htm) , which provides communication between relational and non-relational systems through a common set of components . **ADO.NET** consist of a set of Objects that expose data access services to the .NET environment. ADO.NET is designed to be easy to use, and Visual Studio provides several wizards and other features that you can use to generate **ADO.NET** data access code.

**Data Providers and DataSet**

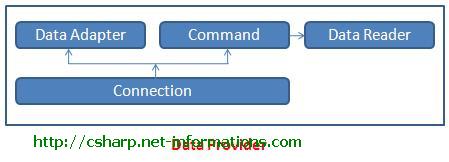


The two key components of ADO.NET are **Data Providers** and **DataSet** . The .Net Framework includes mainly three Data Providers for ADO.NET. They are the Microsoft **SQL Server Data Provider** , **OLEDB Data Provider** and **ODBC Data Provider** . SQL Server uses the SqlConnection object , OLEDB uses the OleDbConnection Object and ODBC uses OdbcConnection Object respectively.

[C# SQL Server Connection](http://csharp.net-informations.com/data-providers/csharp-sql-server-connection.htm)

[C# OLEDB Connection](http://csharp.net-informations.com/data-providers/csharp-oledb-connection.htm)

[C# ODBC Connection](http://csharp.net-informations.com/data-providers/csharp-odbc-connection.htm)



The four Objects from the [.Net Framework](http://vb.net-informations.com/framework/what_is_net_framework.htm) provides the functionality of Data Providers in the ADO.NET. They are **Connection** Object, **Command** Object , **DataReader** Object and **DataAdapter** Object. The Connection Object provides physical connection to the Data Source. The Command Object uses to perform SQL statement or stored procedure to be executed at the Data Source. The DataReader Object is a stream-based , forward-only, read-only retrieval of query results from the Data Source, which do not update the data. Finally the DataAdapter Object , which populate a Dataset Object with results from a Data Source .

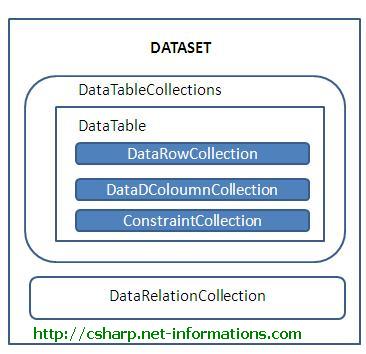
[C# Connection](http://csharp.net-informations.com/data-providers/csharp-ado.net-connection.htm)

[C# Command](http://csharp.net-informations.com/data-providers/csharp-ado.net-Command.htm)

[C# DataReader](http://csharp.net-informations.com/data-providers/csharp-datareader.htm)

[C# DataAdapter](http://csharp.net-informations.com/data-providers/csharp-dataadapter.htm)

**DataSet**



**DataSet** provides a disconnected representation of result sets from the Data Source, and it is completely independent from the Data Source. DataSet provides much greater flexibility when dealing with related Result Sets.

**DataSet** consists of a collection of **DataTable** objects that you can relate to each other with DataRelation objects. The DataTable contains a collection of **DataRow** and **DataCoulumn** Object which contains Data. The DataAdapter Object provides a bridge between the DataSet and the Data Source. From the following section you can see each of the ADO.NET components in details with **C# Source Code** .

# **C# SQl Server Connection**

The SqlConnection Object is Handling the part of physical communication between the C# application and the **SQL Server Database** . An instance of the SqlConnection class in C# is supported the Data Provider for SQL Server Database. The SqlConnection instance takes Connection String as argument and pass the value to the Constructor statement.

**connetionString="Data Source=ServerName;**

**Initial Catalog=DatabaseName;User ID=UserName;Password=Password"**

**cnn = new SqlConnection(connetionString);**

When the connection is established , SQL Commands will execute with the help of the Connection Object and retrieve or manipulate the data in the database. Once the Database activities is over , Connection should be closed and release the Data Source resources .

**cnn.Close();**

The Close() method in SqlConnection Class is used to close the Database Connection. The Close method rolls back any pending transactions and releases the Connection from the SQL Server Database.

# **C# OLEDB Connection**

The C# **OleDbConnection** instance takes Connection String as argument and pass the value to the Constructor statement. An instance of the C# OleDbConnection class is supported the **OLEDB Data Provider** .

**connetionString = "Provider=Microsoft.Jet.OLEDB.4.0;**

**Data Source=yourdatabasename.mdb;";**

**cnn = new OleDbConnection(connetionString);**

When the connection is established between C# application and the specified Data Source, SQL Commands will execute with the help of the **Connection Object** and retrieve or manipulate data in the database. Once the Database activities is over Connection should be closed and release from the data source resources .

**cnn.Close();**

The Close() method in the OleDbConnection class is used to close the Database Connection. The Close method **Rolls Back** any pending transactions and releases the Connection from the Database connected by the OLEDB Data Provider.

# **C# ODBC Connection**

An instance of the **OdbcConnection Class** in C# is supported the ODBC Data Provider. The OdbcConnection instance takes Connection String as argument and pass the value to the Constructor statement. When the connection is established between C# application and the Data Source the SQL Commands will execute with the help of the **Connection Object** and retrieve or manipulate data in the database.

**connetionString = "Driver={Microsoft Access Driver (\*.mdb)};**

**DBQ=yourdatabasename.mdb;";**

**cnn = new OdbcConnection(connetionString);**

Once the Database activities is over you should be closed the Connection and release the Data Source resources . The Close() method in **OdbcConnection Class** is used to close the Database Connection.

**cnn.Close();**

The Close method rolls back any pending transactions and releases the Connection from the Database connected by the **ODBC Data Provider** .

**What is Normalization? And Dinormalization?**

It is eliminate data redundancy and removing duplication data from relation table

1NF (FIRST NORMAL FORM)

No two rows of data must contain repeating group information

**Defination :** Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored. There are several benefits for using Normalization in Database.

**Benefits :**

1. Eliminate data redundancy
2. Improve performance
3. Query optimization
4. Faster update due to less number of columns in one table
5. Index improvement

**What is Virtual Function?**

The concept of the virtual function solves the following problem:

In object-oriented programming when a derived class inherits from a base class, an object of the derived class may be referred to via a pointer or reference of the base class type instead of the derived class type. If there are base class methods overridden by the derived class, the method actually called by such a reference or pointer can be bound either 'early' (by the compiler), according to the declared type of the pointer or reference, or 'late' (i.e. by the runtime system of the language), according to the actual type of the object referred to.

Virtual functions are resolved 'late'. If the function in question is 'virtual' in the base class, the most-derived class's implementation of the function is called according to the actual type of the object referred to, regardless of the declared type of the pointer or reference. If it is not 'virtual', the method is resolved 'early' and the function called is selected according to the declared type of the pointer or reference.

Virtual functions allow a program to call methods that don't necessarily even exist at the moment the code is compiled.

In C++, *virtual methods* are declared by prepending the virtual keyword to the function's declaration in the base class. This modifier is inherited by all implementations of that method in derived classes, meaning that they can continue to over-ride each other and be late-bound.

<http://en.wikipedia.org/wiki/Virtual_function>

### [**Explain the life cycle of an ASP .NET page.**](http://www.dotnetspark.com/qa/234-explain-life-cycle-asp-net-page.aspx)

Life cycle of ASP.Net Web Form   
Page Request >> Start >> Page Init >> Page Load >> Validation >>   
PostBack Event Handling >> Page Rendering >> Page Unload   
Page Request - When the page is requested ASP.Net determines   
whether the page is to be parsed and compiled or a cached verion   
of the page is to be sent without running the page.   
Start - Page propertied REQUEST and RESPONSE are SET, if the   
page is pastback request then the IsPostBack property is SET and   
in addition to this UICulture property is also SET.   
Page Initilization - In this the UniqueID of each property is SET.   
If the request was postback the data is not yet loaded from the   
viewstate.   
Page Load - If it was a postback request then the data gets loaded   
in the control from the ViewState and control property are set.   
Validation - If any control validation present, they are performed   
and IsValid property is SET for each control.   
PostBack Event Handling - If it was a postback request then any   
event handlers are called.   
Page Rendering - Before this the viewstate is saved from the page   
and RENDER method of each page is called.   
Page Unload - Page is fully rendered and sent to the client([Browser](http://www.dotnetspark.com/qa/234-explain-life-cycle-asp-net-page.aspx))   
and is discarded. Page property RESPONSE and REQUEST are unloaded

### **W**[**hat is use of Trim() function and explain different variation ?**](http://www.dotnetspark.com/qa/3958-what-is-use-trim-function-and-explain-different.aspx)

Trim function:  
remove off Space character from the Beginning and from Ending of the instance.  
  
different Variation in the Trim function:  
1)Trim() 2)TrimStart() 3)TrimEnd()  
  
1) Trim():  
Simply says, remove off spacing from the Start and End of the Instance or string.  
2) TrimStart():  
Simply says, Remove off spacing from the Start of the Instance or String   
2) TrimEnd():  
Simply says, Remove off spacing from the End of the Instance or String

**Type Conversion in the .NET**

## 1.Implicit conversion

int smallnum = 654667;

// Implicit conversion

long bigNum = smallnum;

## 2.Explicit conversion

long bigNum = 654667;

// Explicit conversion

int smallnum = (int)bigNum;

## what is XML?

## Extensible Markup Language (XML) strores and transports data. If we use a XML file to store the data then we can do operations with the XML file directly without using the database. The XML format is supported for all applications. It is independent of all software applications and it is accessible by all applications. Here is an example that shows how to insert the data into the XML file and how to retrieve the data from the XML file; also how to bind the data into tha DataList using ASP.NET. For this the following procedure ca

## http://www.c-sharpcorner.com/UploadFile/c25b6d/working-with-xml-file-using-Asp-Net/n be used.