Learn ASP.Net in 1 Day

By Krishna Rungta

Table Of Content

Chapter 7: Insert, Update, Delete: ASP.NET Database Connection Tutorial

1. Fundamentals of Database connectivity
2. ASP.NET Database Connections
3. ASP.NET Read Database using SqlDataReader
4. Insert Database Record using InsertCommand
5. Update Database Record using UpdateCommand
6. Delete Database Record using DeleteCommand
7. Connecting Asp.net Controls to Data

Chapter 8: Asp.Net Page Level Tracing, Debugging, Error Handling [Example]

1. What is Debugging in ASP.NET?
2. What is Tracing in ASP.NET?
3. Page Level Tracing
4. Error Handling: Displaying a Custom Error Page
5. ASP.NET Unhandled Exception
6. ASP.NET Error logging

Chapter 9: How to Host a Website on IIS: Setup & Deploy Web Application

1. How to Download and Install IIS
2. How to Deploy Website in IIS via File copy
3. How to Publish ASP.NET Website

Chapter 10: UNIT TESTING in Asp.Net: Complete Tutorial

1. Introduction to testing for ASP.Net
2. Creating a .NET Unit Testing Project
3. Running the Test Project

Chapter 11: ASP.NET MVC Tutorial for Beginners

1. What is ASP.NET MVC?
2. Why ASP.net MVC?
3. Version History of MVC

4. Features of MVC
5. Thins to remember while creating MVC Application
6. MVC architectural Pattern
7. Web Forms vs. MVC
8. Advantages of ASP.NET MVC
9. Disadvantages of ASP.NET MVC
10. Best practices while using ASP.Net MVC

Chapter 1: What is ASP.NET? and it's ARCHITECTURE

What is ASP.Net?

ASP.Net is a web development platform provided by Microsoft. It is used for creating web-based applications. ASP.Net was first released in the year 2002.

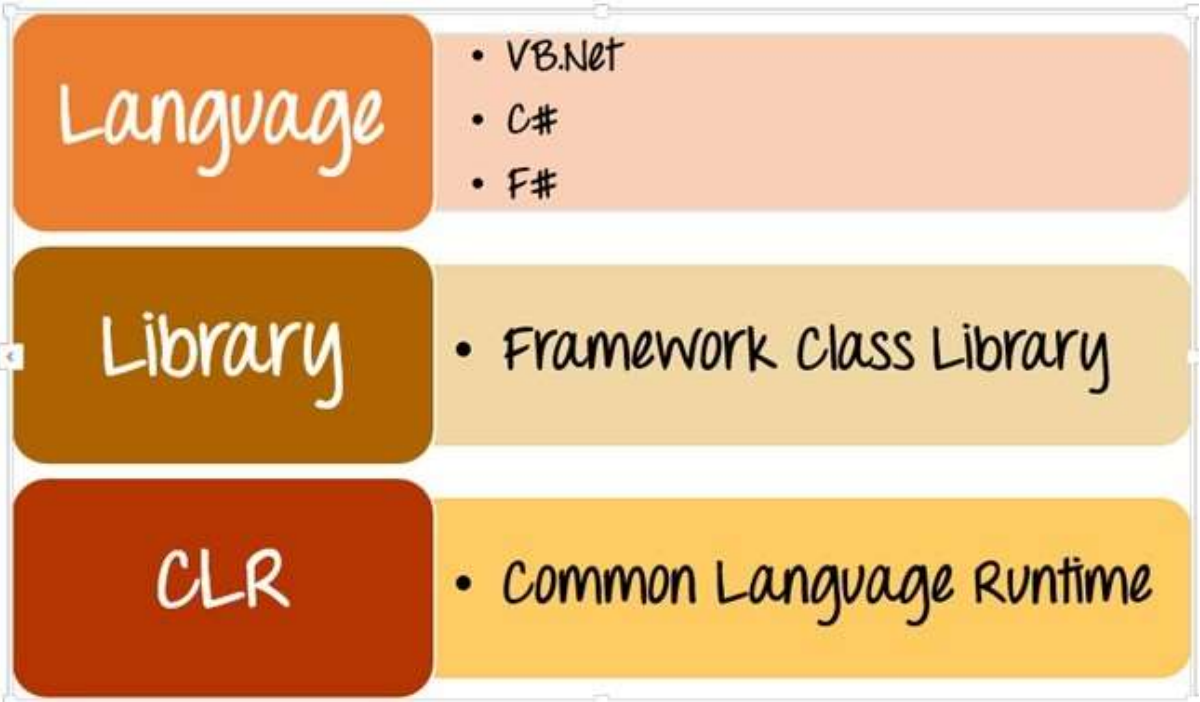
The first version of ASP.Net deployed was 1.0. The most recent version of ASP.Net is version 4.6. ASP.Net is designed to work with the HTTP protocol. This is the standard protocol used across all web applications.

ASP.Net applications can also be written in a variety of .Net languages. These include C#, VB.Net, and J#. In this chapter, you will see some basic fundamental of the .Net framework.

The full form of ASP is Active Server Pages, and .NET is Network Enabled Technologies.

ASP.NET Architecture and its Components

ASP.Net is a framework which is used to develop a Web-based application. The basic architecture of the ASP.Net framework is as shown below.



The architecture of the.Net framework is based on the following key components

1. **Language** – A variety of languages exists for .net framework. They are VB.net and C#. These can be used to develop web applications.
2. **Library** - The .NET Framework includes a set of standard class libraries. The most common library used for web applications in .net is the Web library. The web library has all the necessary components used to develop.Net web-based applications.
3. **Common Language Runtime** - The Common Language Infrastructure or CLI is a platform. .Net programs are executed on this platform. The CLR is used for performing key activities. Activities include Exception handling and Garbage collection.

Below are some of the key characteristics of the ASP.Net framework

1. **Code Behind Mode** – This is the concept of separation of design and code. By making this separation, it becomes easier to maintain the ASP.Net application. The general file type of an ASP.Net file is aspx. Assume we have a web page called MyPage.aspx. There will be another file called MyPage.aspx.cs

which would denote the code part of the page. So Visual Studio creates separate files for each web page, one for the design part and the other for the code.

2. **State Management** – ASP.Net has the facility to control state management. HTTP is known as a stateless protocol. Let's take an example of a shopping cart application. Now, when a user decides what he wants to buy from the site, he will press the submit button.

   The application needs to remember the items the user choose for the purchase. This is known as remembering the state of an application at a current point in time. HTTP is a stateless protocol. When the user goes to the purchase page, HTTP will not store the information on the cart items. Additional coding needs to be done to ensure that the cart items can be carried forward to the purchase page. Such an implementation can become complex at times. But ASP.Net can do state management on your behalf. So ASP.Net can remember the cart items and pass it over to the purchase page.

3. **Caching** – ASP.Net can implement the concept of Caching. This improve's the performance of the application. By caching those pages which are often requested by the user can be stored in a temporary location. These pages can be retrieved faster and better responses can be sent to the user. So caching can significantly improve the performance of an application.

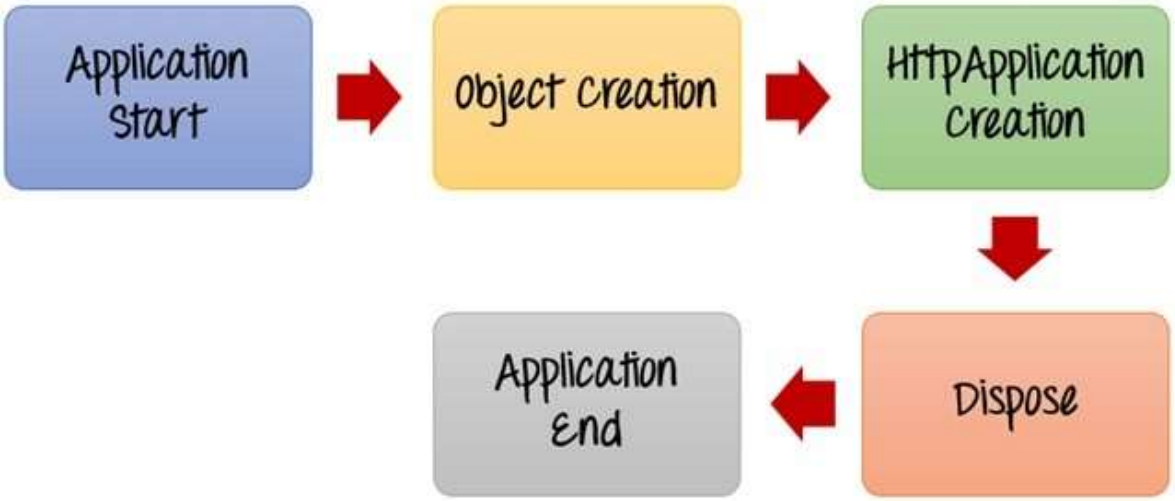
ASP.Net is a development language used for constructing web-based applications. ASP.Net is designed to work with the standard HTTP protocol.

Chapter 2: ASP.NET Application & PAGE Life Cycle

What is ASP.Net Lifecycle?

When an ASP.Net application is launched, there are series of steps which are carried out. These series of steps make up the lifecycle of the application.

Let's look at the various stages of a typical page lifecycle of an ASP.Net Web Application.



ASP.Net Lifecycle

1) **Application Start** - The life cycle of an ASP.NET application starts when a request is made by a user. This request is to the Web server for the ASP.Net Application. This happens when the first user normally goes to the home page for the application for the first time. During this time, there is a method called Application_start which is executed by the web server. Usually, in this method, all global variables are set to their default values.

2) **Object creation** - The next stage is the creation of the HttpContext, HttpRequest & HttpResponse by the web server. The

HttpContext is just the container for the HttpRequest and HttpResponse objects. The HttpRequest object contains information about the current request, including cookies and browser information. The HttpResponse object contains the response that is sent to the client.

3) **HttpApplication creation** - This object is created by the web server. It is this object that is used to process each subsequent request sent to the application. For example, let's assume we have 2 web applications. One is a shopping cart application, and the other is a news website. For each application, we would have 2 HttpApplication objects created. Any further requests to each website would be processed by each HttpApplication respectively.

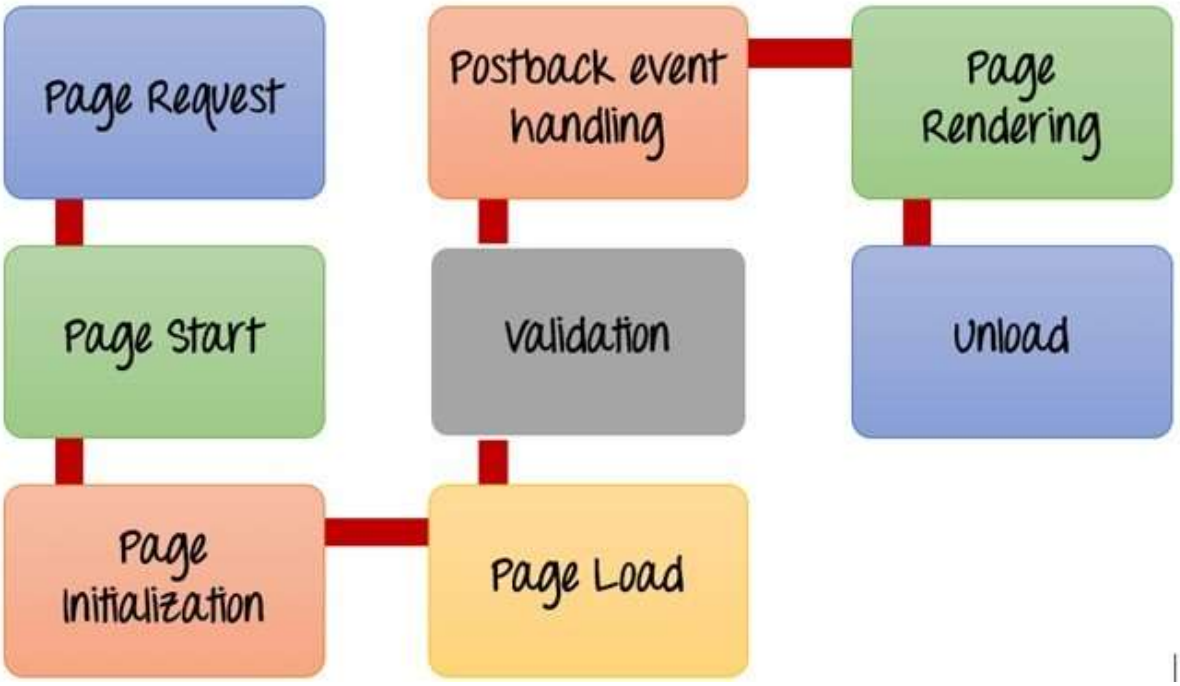
4) **Dispose** - This event is called before the application instance is destroyed. During this time, one can use this method to manually release any unmanaged resources.

5) **Application End** - This is the final part of the application. In this part, the application is finally unloaded from memory.

What is ASP.Net Page Lifecycle?

When an ASP.Net page is called, it goes through a particular lifecycle. This is done before the response is sent to the user. There are series of steps which are followed for the processing of an ASP.Net page.

Let's look at the various stages of the lifecycle of an ASP.Net web page.



ASP.Net Page Lifecycle

1. **Page Request**- This is when the page is first requested from the server. When the page is requested, the server checks if it is requested for the first time. If so, then it needs to compile the page, parse the response and send it across to the user. If it is not the first time the page is requested, the cache is checked to see if the page output exists. If so, that response is sent to the user.
2. **Page Start** – During this time, 2 objects, known as the Request and Response object are created. The Request object is used to hold all the information which was sent when the page was requested. The Response object is used to hold the information which is sent back to the user.
3. **Page Initialization** – During this time, all the controls on a web page is initialized. So if you have any label, textbox or any other controls on the web form, they are all initialized.
4. **Page Load** – This is when the page is actually loaded with all the default values. So if a textbox is supposed to have a default value, that value is loaded during the page load time.
5. **Validation** – Sometimes there can be some validation set on the form. For example, there can be a validation which says that a list

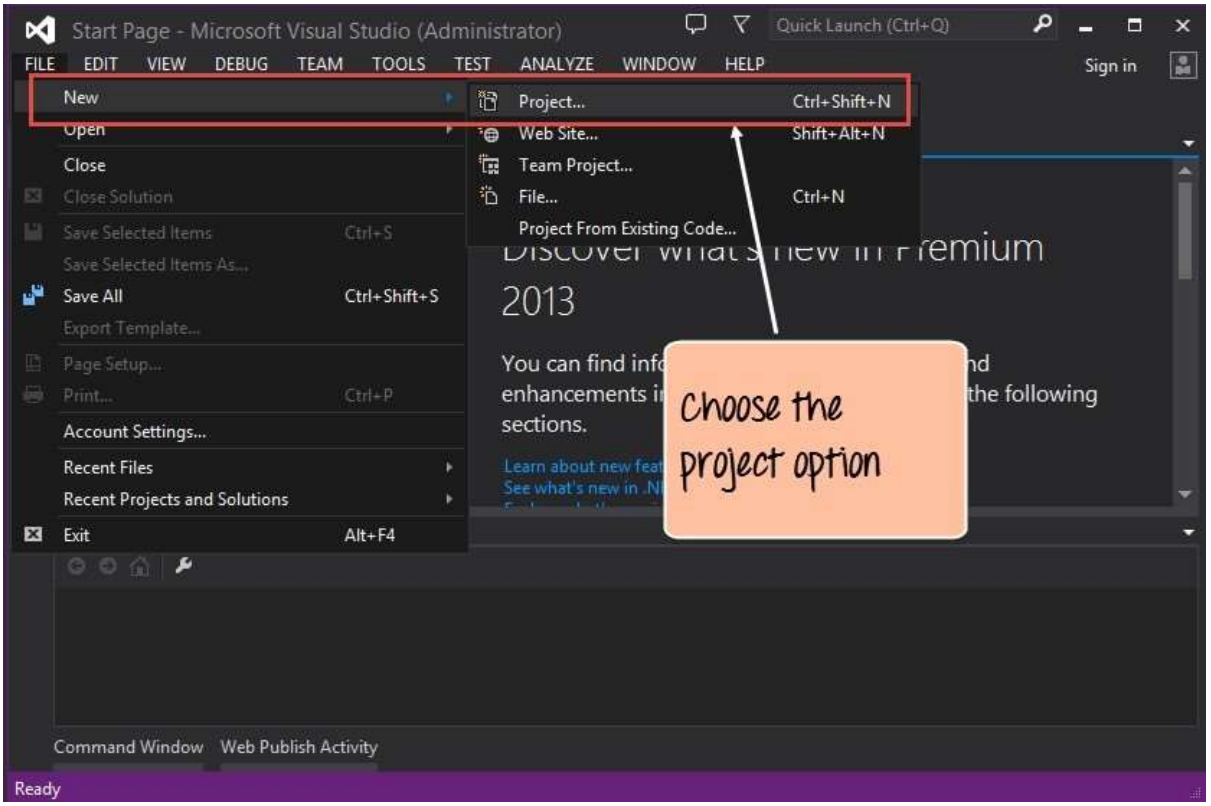
box should have a certain set of values. If the condition is false, then there should be an error in loading the page.

6. **Postback event handling** – This event is triggered if the same page is being loaded again. This happens in response to an earlier event. Sometimes there can be a situation that a user clicks on a submit button on the page. In this case, the same page is displayed again. In such a case, the Postback event handler is called.
7. **Page Rendering** – This happens just before all the response information is sent to the user. All the information on the form is saved, and the result is sent to the user as a complete web page.
8. **Unload** – Once the page output is sent to the user, there is no need to keep the ASP.net web form objects in memory. So the unloading process involves removing all unwanted objects from memory.

Chapter 3: ASP.NET First Program
Example: Hello World

Let's look at an example of how we can implement a simple "hello world" application. For this, we would need to implement the below- mentioned steps.

Step 1) The first step involves the creation of a new project in Visual Studio. After launching Visual Studio, you need to choose the menu option New->Project.

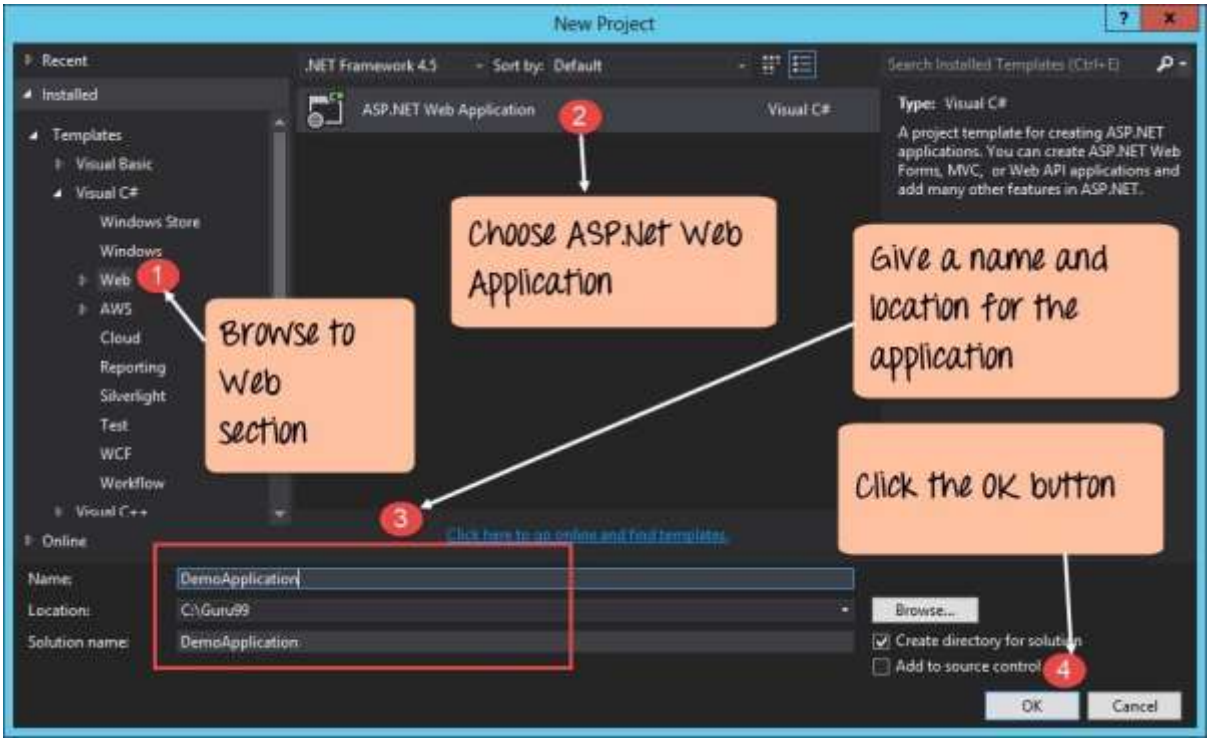


Step 2) The next step is to choose the project type as an ASP.Net Web application. Here we also need to mention the name and location of our project.

1. In the project dialog box, you can see various options for creating different types of projects. Click the Web option on the left-hand

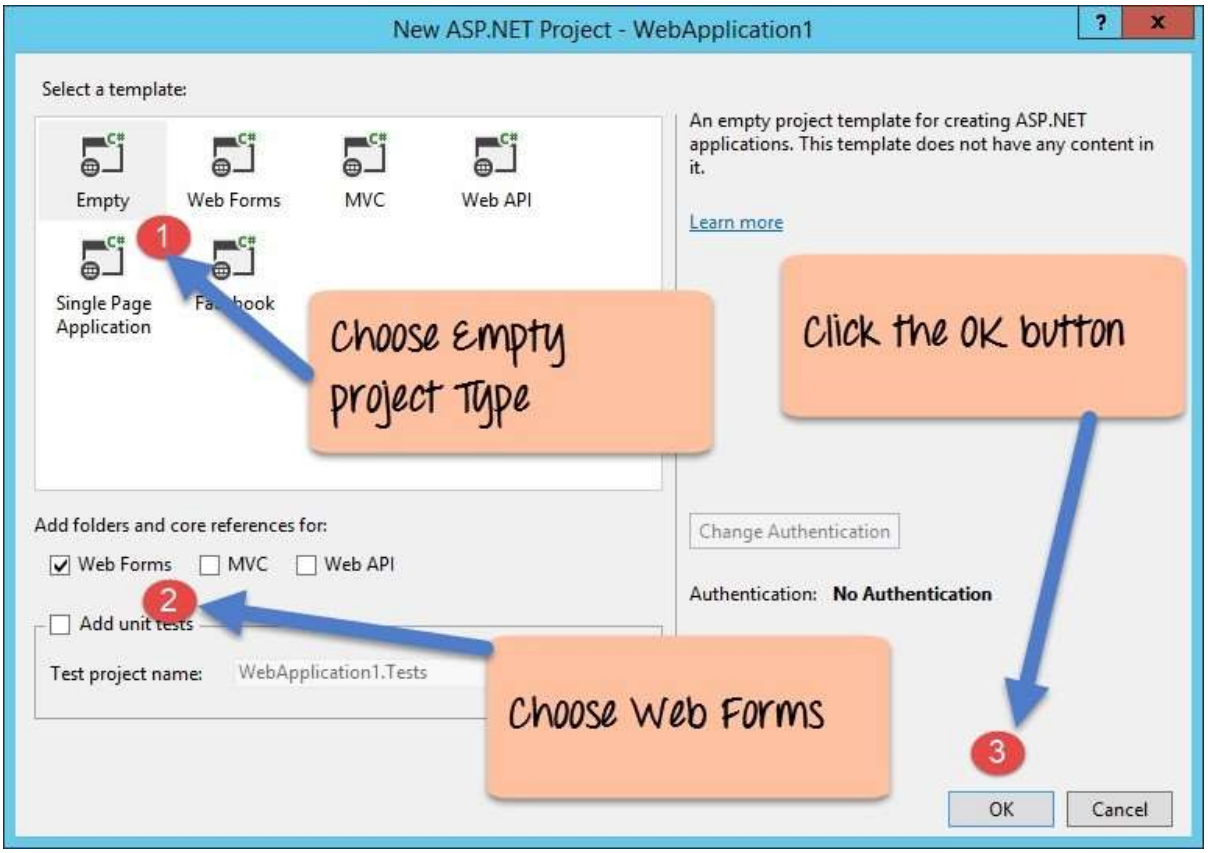
side.

2. When we click the Web option in the previous step, we will be able to see an option for ASP.Net Web Application. Click this option.
3. We then give a name for the application, which in our case is DemoApplication. We also need to provide a location to store our application.
4. Finally, we click the 'OK' button to let Visual Studio to create our project.



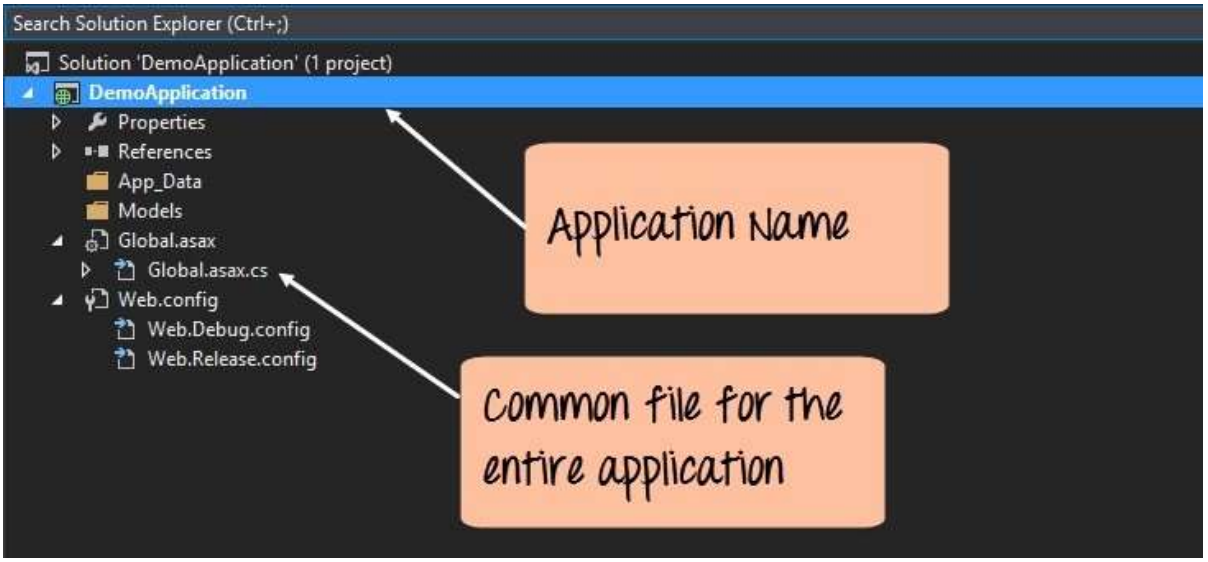
Step 3) In the next screen, you have to choose the type of ASP.net web application that needs to be created. In our case, we are going to create a simple Web Form application.

1. First, choose the project type as 'Empty'. This will ensure that we start with a basic application which is simple tounderstand.
2. We choose the option "web Forms". This adds the basic folders. These are required for a basic Web Forms Application.
3. Finally, we click the 'OK' button to allow Visual Studio to create our application.



If the above steps are followed, you will get the below output in Visual Studio.

Output:-

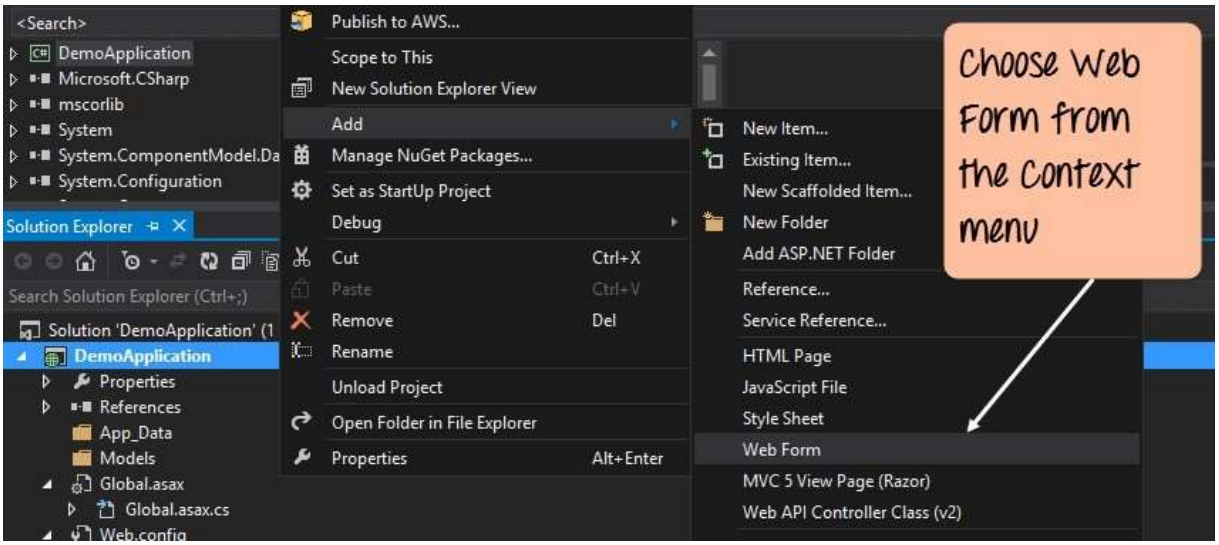


In the Solution Explorer, you will be able to see the DemoApplication Solution. This solution will contain 2 project files as shown above. At the moment, one of the key files in the project is the '**Global.asax.cs'**. This file contains application specific information. In this file, you

would initialize all application specific variables to their default values.

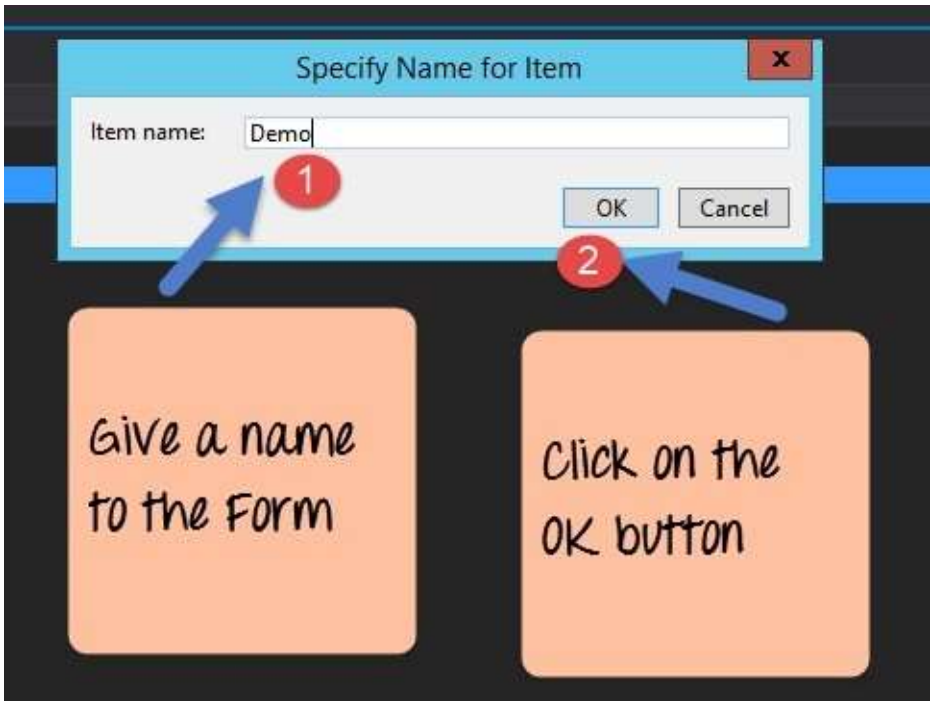
Step 4) Now, it's time to add a Web Form file to the project. This is the file which will contain all the web-specific code for our project.

- Right-click on the DemoApplication project and
- Choose Add->Web Form from the context menu.



Step 5) In the next screen we are going to be prompted to provide a name for the web form.

1. Give a name for the Web Form. In our case, we are giving it a name of Demo.
2. Click the Ok button.



Automatically Visual Studio will create the Demo Web Form and will open it in Visual Studio.

Step 6) The next step is to add the code, which will do the work of displaying “Hello World.” This can be done by just adding one line of code to the Demo.aspx file.


```
<html xmlns="www.w3.org/1999/xhtml">
<head runat="server">
        <title></title>
</head>
<body>
        <form id="form1" runat="server">
        <div>
```

```
        <%Response. Write( "HeIIo World") %>

        </div>
        </form>
</body>
</html>
```

Code Explanation:-

- The Response object in ASP.Net is used to send information back to the user. So in our case, we are using the method “Write” of the Response object to write the text “Hello World.” The <% and %> markers are used to add ASP.net specific code.

If you follow all of the above steps and run your program in Visual Studio, you will get the following output.

Output:-

From the output, you can clearly see that ‘Hello World’ was displayed in the browser.