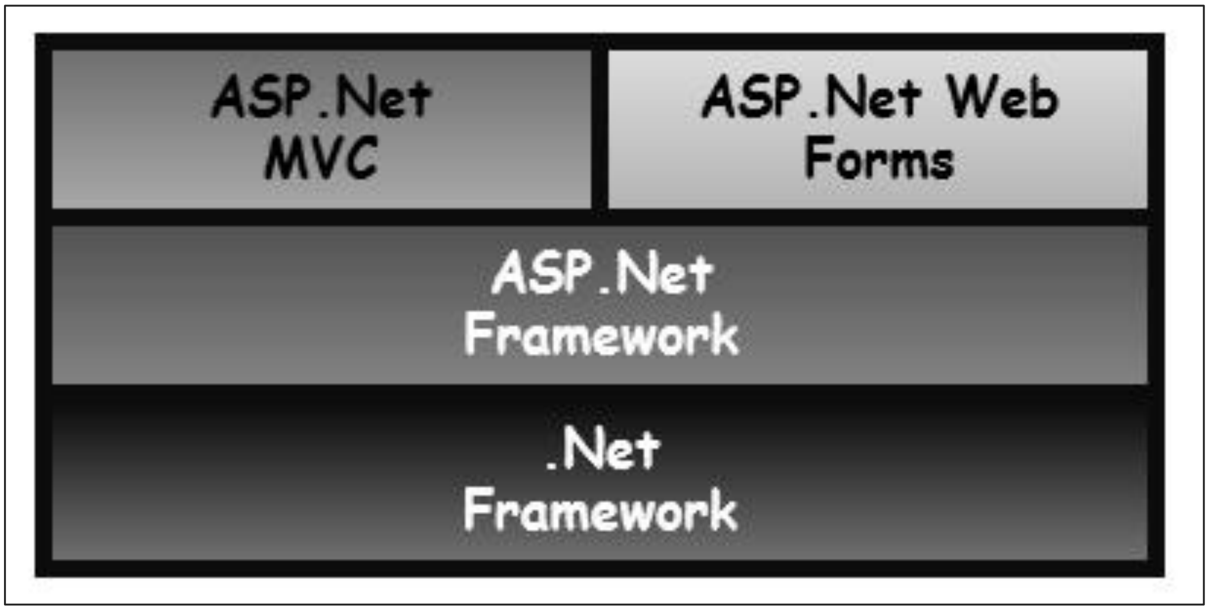
1. ASP.NET MVC – OVERVIEW

ASP.NET MVC is basically a web development framework from Microsoft, which combines the features of MVC (Model-View-Controller) architecture, the most up-to-date ideas and techniques from Agile development, and the best parts of the existing ASP.NET platform.

ASP.NET MVC is not something, which is built from ground zero. It is a complete alternative to traditional ASP.NET Web Forms. It is built on the top of ASP.NET, so developers enjoy almost all the ASP.NET features while building the MVC application.



History

ASP.NET 1.0 was released on January 5, 2002, as part of .Net Framework version 1.0. At that time, it was easy to think of ASP.NET and Web Forms as one and the same thing. ASP.NET has however always supported two layers of abstraction:

- **System.Web.UI:** The Web Forms layer, comprising server controls, ViewState, and so on.

- **System.Web:** It supplies the basic web stack, including modules, handlers, the HTTP stack, etc.

By the time ASP.NET MVC was announced in 2007, the MVC pattern was becoming one of the most popular ways of building web frameworks.

In April 2009, the ASP.NET MVC source code was released under the Microsoft Public License (MS-PL). "ASP.NET MVC framework is a lightweight, highly testable presentation framework that is integrated with the existing ASP.NET features.

Some of these integrated features are master pages and membership-based authentication. The MVC framework is defined in the System.Web.Mvc assembly.

In March 2012, Microsoft had released part of its web stack (including ASP.NET MVC, Razor and Web API) under an open source license (Apache License 2.0). ASP.NET Web Forms was not included in this initiative.

Why ASP.NET MVC?

Microsoft decided to create their own MVC framework for building web applications. The MVC framework simply builds on top of ASP.NET. When you are building a web application with ASP.NET MVC, there will be no illusions of state, there will not be such a thing as a page load and no page life cycle at all, etc.

Another design goal for ASP.NET MVC was to be extensible throughout all aspects of the framework. So when we talk about views, views have to be rendered by a particular type of view engine. The default view engine is still something that can take an ASPX file. But if you don't like using ASPX files, you can use something else and plug in your own view engine.

There is a component inside the MVC framework that will instantiate your controllers. You might not like the way that the MVC framework instantiates your controller, you might want to handle that job yourself. So, there are lots of places in MVC where you can inject your own custom logic to handle tasks.

The whole idea behind using the Model View Controller design pattern is that you maintain a separation of concerns. Your controller is no longer encumbered with a lot of ties to the ASP.NET runtime or ties to the ASPX page, which is very hard to test. You now just have a class with regular methods on it that you can invoke in unit tests to find out if that controller is going to behave correctly.

Benefits of ASP.NET MVC

Following are the benefits of using ASP.NET MVC:

- Makes it easier to manage complexity by dividing an application into the model, the view, and the controller.

- Enables full control over the rendered HTML and provides a clean separation of concerns.

- Direct control over HTML also means better accessibility for implementing compliance with evolving Web standards.

- Facilitates adding more interactivity and responsiveness to existing apps.

- Provides better support for test-driven development (TDD).
- Works well for Web applications that are supported by large teams of developers and for Web designers who need a high degree of control over the application behavior.