| **ASP.NET Web Forms** | **ASP.NET MVC** |
| --- | --- |
| ASP.NET Web Forms use Page controller pattern approach for rendering layout. In this approach, every page has its own controller, i.e., code-behind file that processes the request. | ASP.NET MVC uses Front Controller approach. That approach means a common controller for all pages processes the requests. |
| No separation of concerns. As we discussed that every page (*.aspx*) has its own controller (code behind i.e. *aspx.cs/.vb* file), so both are tightly coupled. | Very clean separation of concerns. View and Controller are neatly separate. |
| Because of this coupled behavior, automated testing is really difficult. | Testability is a key feature in ASP.NET MVC. Test driven development is quite simple using this approach. |
| In order to achieve stateful behavior, viewstate is used. Purpose was to give developers the same experience of a typical WinForms application. | ASP.NET MVC approach is stateless as that of the web. So there is no concept of viewstate. |
| Statefulness has a lots of problem for web environment in case of excessively large viewstate. Large viewstate means increase in page size. | As controller and view are not dependent and also no viewstate concept in ASP.NET MVC, so output is very clean. |
| ASP.NET WebForms model follows a Page Life cycle. | No Page Life cycle like WebForms. Request cycle is simple in ASP.NET MVC model. |
| Along with statefulness, Microsoft tries to introduce server-side controls as in Windows applications. Purpose was to provide somehow an abstraction to the details of HTML. In ASP.NET Web Forms, minimal knowledge of HTML, JavaScript and CSS is required. | In MVC, detailed knowledge of HTML, JavaScript and CSS is required. |
| The above abstraction was good but provides limited control over HTML, JavaScript and CSS which is necessary in many cases. | Full control over HTML, JavaScript and CSS. |
| With a lots of control libraries availability and limited knowledge of other related technologies, ASP.NET WebForms is RAD(Rapid Application Development) approach. | It's a step back. For developers decrease in productivity. |
| It’s good for small scale applications with limited team size. | It’s better as well as recommended approach for large-scale applications where different teams are working together. |