MVC FRAMEWORK

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

As per the official definition, **Model-View-Controller (MVC)** is a software architectural pattern for implementing user interfaces. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

Audience

This tutorial is targeted for .NET programmers beginning to learn MVC framework. This tutorial will bring you to intermediate level of knowledge in MVC, covering all the important aspects of MVC Framework with complete hands-on code experience.

Prerequisites

Before proceeding with this tutorial, we assume the readers have a basic knowledge of ASP.NET development (C# and VB language) and Visual Studio software installed on their system.

Disclaimer & Copyright

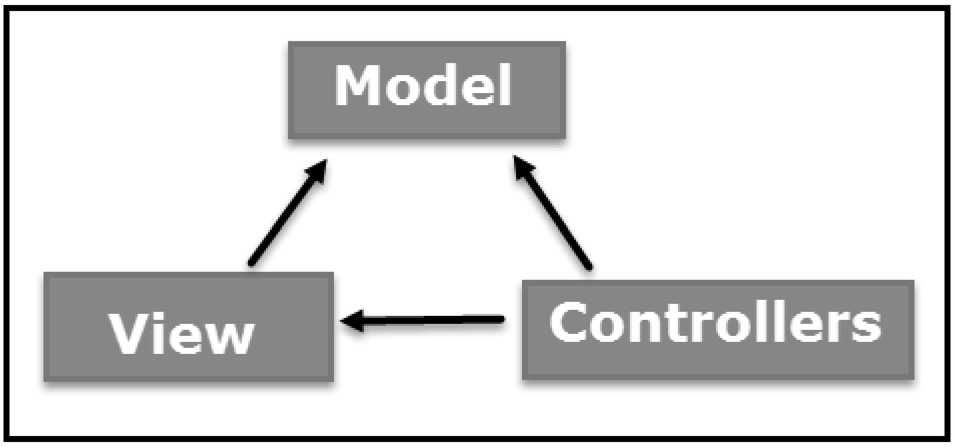
Table of Contents

1. MVC Framework – Introduction

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the **view**, and the **controller**. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

MVC Components

Following are the components of MVC:



Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

ASP.NET MVC

ASP.NET supports three major development models: Web Pages, Web Forms and MVC (Model View Controller). ASP.NET MVC framework is a lightweight, highly testable presentation framework that is integrated with the existing ASP.NET features, such as master pages, authentication, etc. Within .NET, this framework is defined in the System.Web.Mvc assembly. The latest version of the MVC Framework is 5.0. We use Visual Studio to create ASP.NET MVC applications which can be added as a template in Visual Studio.

ASP.NET MVC Features

ASP.NET MVC provides the following features:

- Ideal for developing complex but lightweight applications.

- Provides an extensible and pluggable framework, which can be easily replaced and customized. For example, if you do not wish to use the in-built Razor or ASPX View Engine, then you can use any other third-party view engines or even customize the existing ones.

- Utilizes the component-based design of the application by logically dividing it into Model, View, and Controller components. This enables the developers to manage the complexity of large-scale projects and work on individual components.

- MVC structure enhances the test-driven development and testability of the application, since all the components can be designed interface-based and tested using mock objects. Hence, ASP.NET MVC Framework is ideal for projects with large team of web developers.

- Supports all the existing vast ASP.NET functionalities, such as Authorization and Authentication, Master Pages, Data Binding, User Controls, Memberships, ASP.NET Routing, etc.

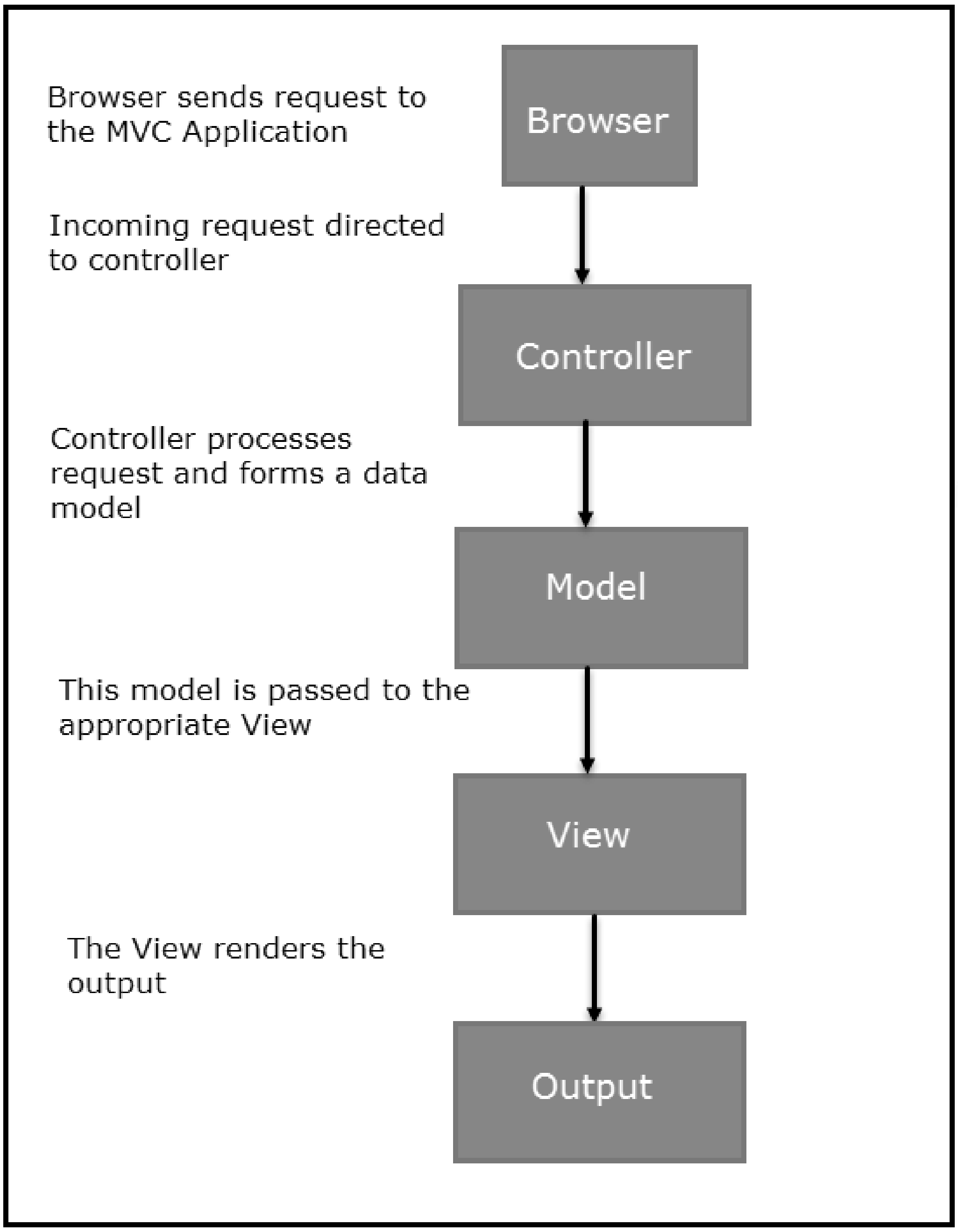
- Does not use the concept of View State (which is present in ASP.NET). This helps in building applications, which are lightweight and gives full control to the developers.

Thus, you can consider MVC Framework as a major framework built on top of ASP.NET providing a large set of added functionality focusing on component-based development and testing.

2. MVC Framework – Architecture

In the last chapter, we studied the high-level architecture flow of MVC Framework. Now let us take a look at how the execution of an MVC application takes place when there is a certain request from the client. The following diagram illustrates the flow.

MVC Flow Diagram


Browser sends request to the MVC Application
Browser
Incoming request directed to controller
Controller
Controller processes request and forms a data model
Model
This model is passed to the appropriate View
View
The View renders the output
Output

Flow Steps

Step 1: The client browser sends request to the MVC Application.

Step 2: Global.ascx receives this request and performs routing based on the URL of the incoming request using the RouteTable, RouteData, UrlRoutingModule and MvcRouteHandler objects.

Step 3: This routing operation calls the appropriate controller and executes it using the IControllerFactory object and MvcHandler object's Execute method.

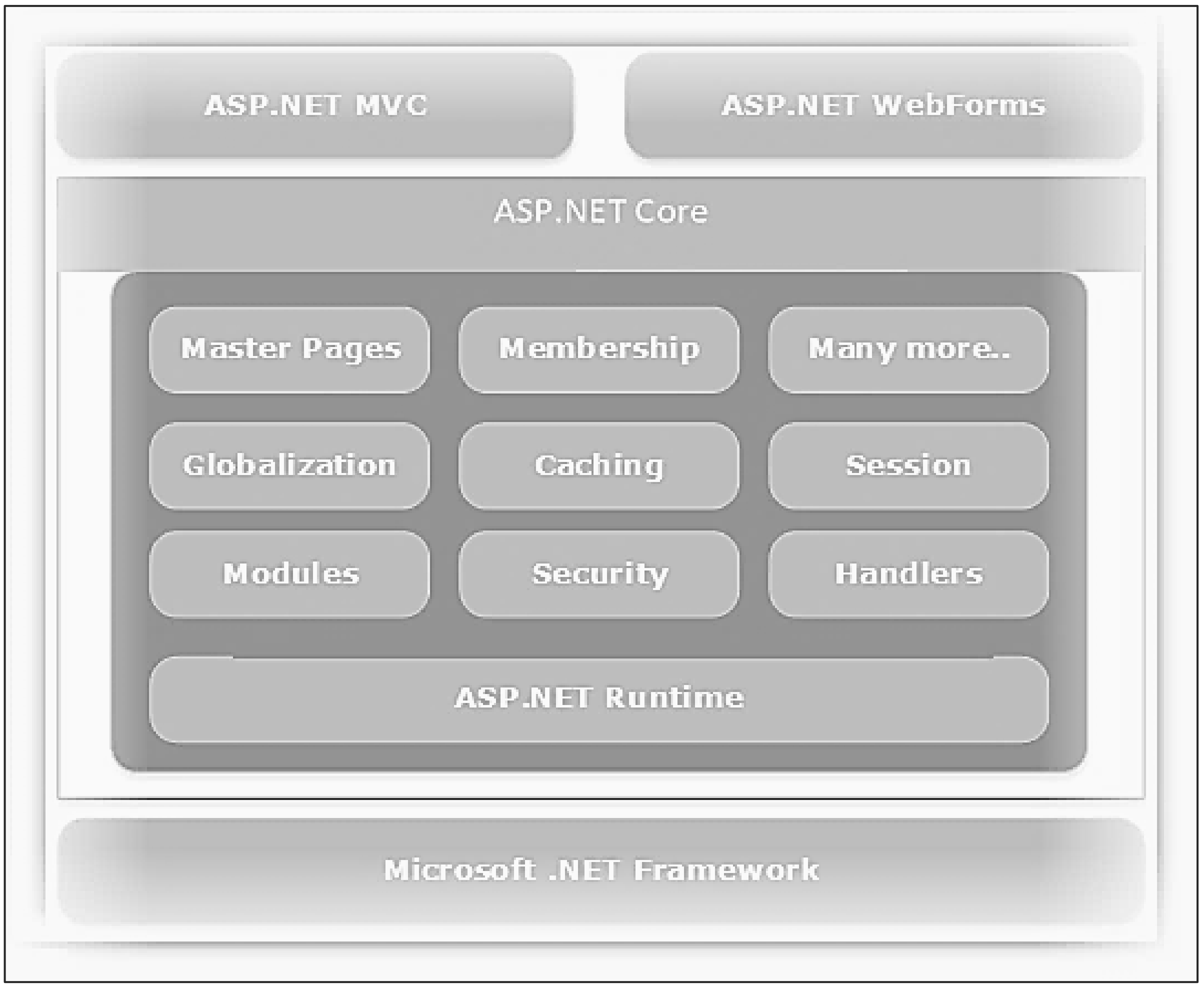
Step 4: The Controller processes the data using Model and invokes the appropriate method using ControllerActionInvoker object

Step 5: The processed Model is then passed to the View, which in turn renders the final output.

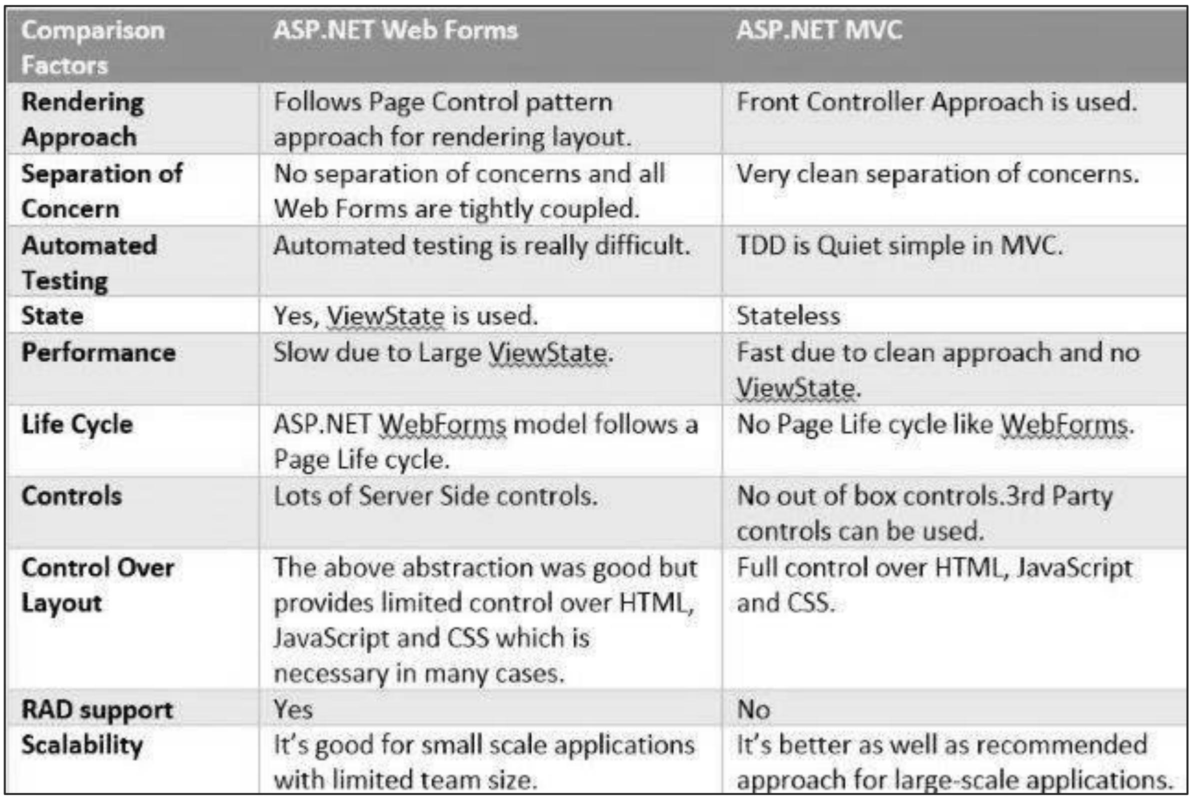
3. MVC Framework – ASP.NET Forms

MVC and ASP.NET Web Forms are inter-related yet different models of development, depending on the requirement of the application and other factors. At a high level, you can consider that MVC is an advanced and sophisticated web application framework designed with separation of concerns and testability in mind. Both the frameworks have their advantages and disadvantages depending on specific requirements. This concept can be visualized using the following diagram:

MVC and ASP.NET Diagram



Comparison Table



Comparison Factors	ASP.NET Web Forms	ASP.NET MVC
Rendering Approach	Follows Page Control pattern approach for rendering layout.	Front Controller Approach is used.
Separation of Concern	No separation of concerns and all Web Forms are tightly coupled.	Very clean separation of concerns.
Automated Testing	Automated testing is really difficult.	TDD is Quiet simple in MVC.
State	Yes, ViewState is used.	Stateless
Performance	Slow due to Large ViewState.	Fast due to clean approach and no ViewState.
Life Cycle	ASP.NET WebForms model follows a Page Life cycle.	No Page Life cycle like WebForms.
Controls	Lots of Server Side controls.	No out of box controls.3rd Party controls can be used.
Control Over Layout	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.
RAD support	Yes	No
Scalability	It's good for small scale applications with limited team size.	It's better as well as recommended approach for large-scale applications.