MVC Tutorial for Beginners

Published On: September 28, 2024

MVC Tutorial for Beginners

MVC is a framework for back-end web development. It offers an object-oriented approach to code structuring and is used to create server-side web applications. Master the fundamentals with this MVC tutorial d esigned for beginners.

Download MVC Tutorial PDF

Introduction to MVC

MVC is a framework for back-end web development. It offers an object-oriented approach to code structuring and is used to create server-side web applications. We cover the following in this MVC tutorial:

- Overview of MVC
- Environment Setup
- Getting Started with MVC
- Routing in MVC
- MVC Controllers
- MVC Model
- MVC View
- MVC-Related Frameworks
- Advantages of the ASP.NET MVC Framework

Overview of MVC

Model View Controller, or MVC, is a software design paradigm that creates user interfaces. The program logic is split up into three sections:

- **Model:** Information is internally represented as a set of classes that describe business logic.
- **View:** The user interface, created with HTML, CSS, and Jquery, displays and receives data from the user.
- **Controller:** The program handles incoming requests and connects the model and view.

Kickstart your career with our wide range of **software training courses**.

MVC Interview Questions and Answers

Environment Setup

Visual Studio 2012 and later versions come with an MVC development tool. Visual Studio 2010 SP1/Visual Web Developer 2010 Express SP1 can install it. MVC 4 may be installed with Visual Studio 2010 by utilizing the Web Platform Installer.

SQL Server is also included in the free version of Visual Studio that Microsoft offers; you can get it at https://www.visualstudio.com.

Installation

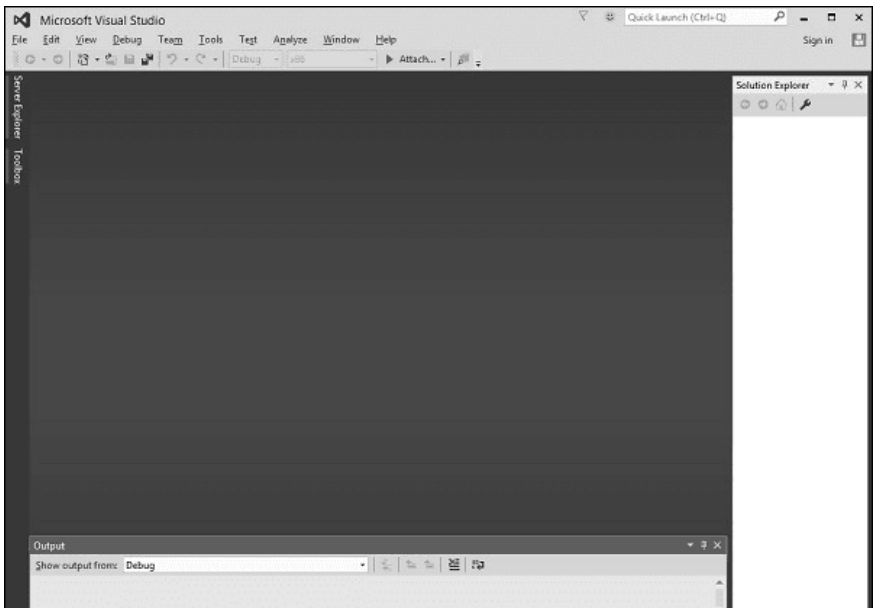
Step 1: Launch the installer after the download is finished. It will show the dialog box that follows.

Step 2: The installation process will begin when you click the "Install" button. Following a successful installation, the following dialog box will appear.

Step 3: Close this dialog box and, if necessary, restart your computer.

Step 4: Select Visual Studio from the Start Menu. The dialog box that appears will open. Just the preparation will take some time the first time.

When everything is finished, Visual Studio's main window will appear, as seen in the screenshot below. It's now time for you to begin your application.



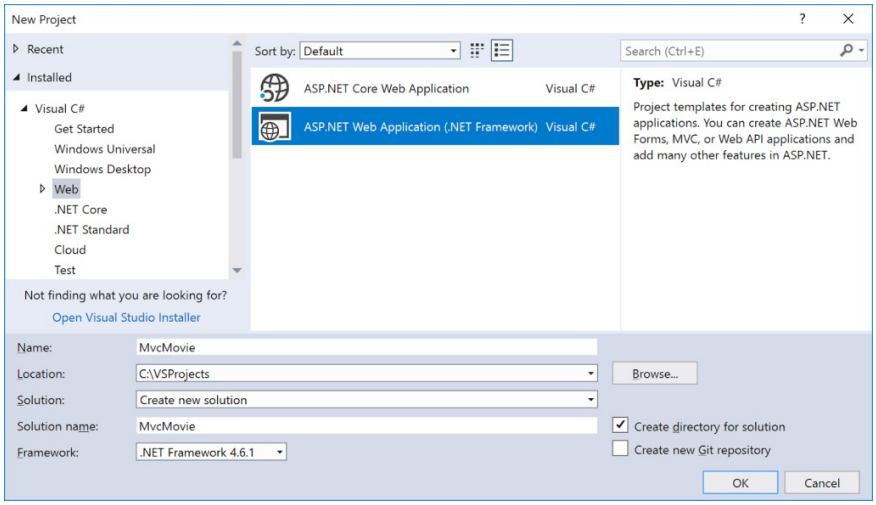
MVC Tutorial 1

Our **ASP.Net training program in Chennai** gives you wings to soar high in your dream career.

MVC Syllabus PDF

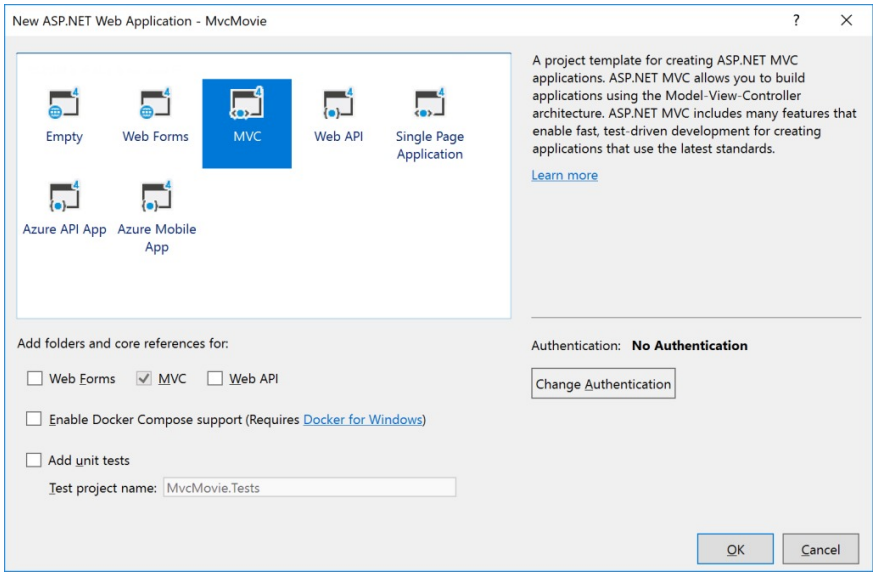
Getting Started with MVC

Go to the Start screen and choose New Project. Choose the Visual C# category on the left, followed by Web, and finally the ASP.NET Web Application (.NET Framework) project template in the New Project dialog box. Create a project named "MvcMovie" and select OK.



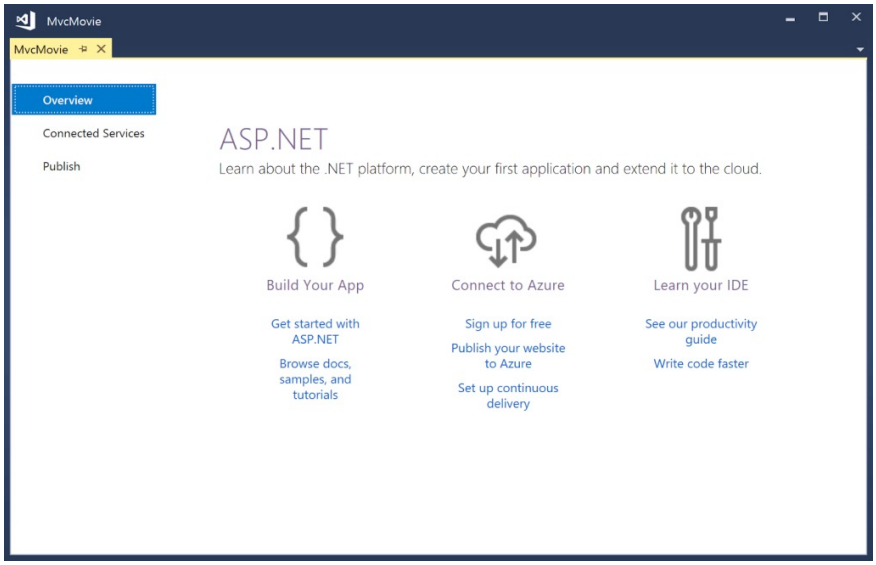
MVC Tutorial 2

Select MVC from the New ASP.NET Web Application dialog box, then click OK.



MVC Tutorial 3

You just started an ASP.NET MVC project in Visual Studio, and it already has a working application! You don't even need to do anything! This is a straightforward "Hello World!" project that might serve as an excellent foundation for your application.



MVC Tutorial 4

- Hit F5 to initiate debugging. Visual Studio launches IIS Express and launches your web application when you hit F5.
- After that, Visual Studio starts a browser and displays the main page of the program. Observe that the browser's address bar displays *localhost:port#* rather than anything like example.com.
- This is because localhost always refers to your local machine, which is currently running the program you just created.
- The web server in a Visual Studio web project uses a random port. The port number in the picture below is 1234.
- The port number that appears will change when you launch the application.

This basic design provides you with the Home, Contact, and About pages right out of the box. The Home, About, and Contact links are not visible in the image below. To view these links, you may need to click the navigation symbol, depending on how big your browser window is.

MVC Tutorial 6

The program helps users log in and register. The next stage is to modify this application's functionality and gain some knowledge about ASP.NET MVC.

Example:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;
```

```
using System.Web.Mvc;

namespace MVCFirstApp.Controllers {

  public class HomeController : Controller {

    // GET: Home

    public string Index(){

      return "Hello World, this is ASP.Net MVC Tutorials";

    }

  }

}
```

MVC Developer Salary in Chennai

MVC Lifecycle

The MVC life cycle consists of a sequence of actions or events that are utilized to process requests and modify the state of an application. The idea behind different framework life cycles may already be familiar to you; MVC is not the only example.

The ASP.NET webforms framework, for instance, has an intricate page life cycle. Different.NET platforms have different application life cycles, such as Windows phone apps. Regardless of the technology, one thing holds for all of these platforms: MVC is no exception when it comes to how properly utilizing the features available requires a grasp of the processing pipeline.

Application Lifecycle

The period from when the application process starts using IIS until it ends is referred to as the application life cycle. The application start and finish events in your application's startup file indicate this.

The Request Lifecycle

Routing is the first entry point for any MVC application. Using the URL Routing Module, the ASP.NET framework determines how to handle a request once it has been received.

Every route has a corresponding route handler, which serves as the MVC framework's entry point. It is the series of actions that our program takes each time it receives an HTTP request.

Modules are. NET elements with additional functionality that may be included in the application life cycle. The incoming URL must match the routes that we specify in our application for the routing module to function.

MVC Tutorial 7

- Converting the route data into a tangible controller that can respond to requests is handled by the MVC framework.
- Action execution is the next important step after creating the controller. To activate the controller, a component known as the action invoker locates and chooses a suitable action method.
- The next step, called Result Execution, begins when our action result is ready. Declaring the result and carrying it out are two different things in MVC.
- The View Engine, which locates and renders our view, will be triggered if the result is a view type.

Explore our courses with job assistance in our **placement training institute in Chennai**.

Routing in MVC

Sending an HTTP request to a controller is known as routing and System.Web.Routing provides the mechanism for this processing. There is no assembly included in ASP.NET MVC. It was formally released as a.NET 3.5 SP1 and is in fact a component of the ASP.NET runtime.

The MVC framework uses a *System.Web.Routing;* however, ASP.NET Dynamic Data also uses it. Routing is used by the MVC framework to route a request to a controller. The section of your application where you declare your application's route is called *Global.asax*.

> *using System;*

```
using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

namespace MVCFirstApp {

  public class MvcApplication:
System.Web.HttpApplication {

    protected void Application_Start(){

      AreaRegistration.RegisterAllAreas();

      RouteConfig.RegisterRoutes(RouteTable.Routes);

    }

  }

}
```

The RouteConfig class implementation, which has one method called RegisterRoutes, is shown below.

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

namespace MVCFirstApp {
```

```
public class RouteConfig {

    public static void RegisterRoutes(RouteCollection
routes){

        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(

          name: "Default",

          url: "{controller}/{action}/{id}",

          defaults: new{ controller = "Home", action =
"Index", id = UrlParameter.Optional});

    }

  }

}
```

Understanding Routes

The ASP.NET routing system is used by MVC applications to determine how URLs correspond to controllers and actions.

To get us started, Visual Studio includes a few default routes when it starts the MVC project. You will notice that Visual Studio has pointed the browser to port 63664 when you execute your application. Visual Studio assigns a random port when the project is created, so you will almost likely see a different port number in the URL that your browser requests.

Since we included a HomeController in the previous example, you may also request any of the following URLs, and they will lead to the HomeController's Index action.

http://localhost:63664/Home/

http://localhost:63664/Home/Index

The result from HomeController's Index function is returned to a browser when it requests *http://mysite/ or http://mysite/Home.*

By altering the URL in the browser, you can give this another go. It is *http://localhost:63664/* in this case, though the port may alter.

The MVC application will display the same result if you add /Home or /Home/Index to the URL and hit the "Enter" button.

Custom Convention

Of course, you are free to add other routes. You can add your own route entries if you'd want to use other action names, have different ID parameters, or simply have a different URL structure for your website overall.

Let's examine a straightforward illustration. Imagine that we have a page with the list of processes on it. The code that will direct users to the process page is as follows:

> *routes.MapRoute(*
>
> *"Process",*

```
    "Process/{action}/{id}",

    defaults: new{

        controller = "Process", action = "List ", id =
UrlParameter.Optional}

    );
```

The Process Controller is the place someone goes when they enter and search for a URL with Process/Action/Id.

A request that comes in now appears to be from localhosts/process. The routing engine will use the default action of List to pass that forward, based on this routing setup.

Here is the whole implementation of the class.

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

namespace MVCFirstApp{

  public class RouteConfig{

    public static void RegisterRoutes(RouteCollection
routes){

      routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

      routes.MapRoute(

        "Process", "Process/{action}/{id}",

        defaults: new{

          controller = " Process", action = "List ", id =
```

```
            UrlParameter.Optional});

        routes.MapRoute(

            name: "Default", url:
"{controller}/{action}/{id}",

            defaults: new{

              controller = "Home", action = "Index", id =

              UrlParameter.Optional});

    }

  }

}
```

Step 1: Execute this and use the following URL to obtain a process page: *http://localhost:63664/Process*

The routing engine is searching for ProcessController, which is not available, so you will see an HTTP 404.

Step 2: Create ProcessController by choosing Add → Controller from the solution explorer when you right-click on the Controllers folder.

MVC Tutorial 11

The Add Scaffold dialog will appear.

MVC Tutorial 12

Step 3: Click the “Add” button after selecting the MVC 5 Controller-Empty option.

You will see the Add Controller dialog box.

MVC Tutorial 13

Step 4: Click the “Add” button after naming the object ProcessController.

The Controllers folder now has a new C# file named

ProcessController.cs, which may be edited in Visual Studio.

We want to have a List action here rather than an Index action because our default action from now on will be List.

Step 5: Use the following code to return some string from this action function and change the return type from ActionResult to string.

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace MVCFirstApp.Controllers{

  public class ProcessController : Controller{

    // GET: Process

    public string List(){

      return “This is Process page”;

    }
```

```
    }

}
```

Step 6: The default route's result will be displayed once more when you start this application. You can view the outcome of the ProcessController by entering the following URL: *http://localhost:63664/Process/List.*

MVC Project Ideas

MVC Controllers

Controllers are C# classes that extend the built-in controller base class, System.Web.Mvc.Controller. Every public method in a controller is referred to as an action method, which means you can call it to act on the Web by providing a URL.

Controllers should be placed in the Controllers folder that Visual Studio generated during the project setup, according to MVC conventions.

Let's create a new ASP.NET MVC project and examine a basic example of a controller.

Step 1: Launch Visual Studio and select File > New > Project from the menu.

A brand-new Project dialog box appears.

Step 2: Choose Templates → Visual C# → Web from the left window.

Step 3: Choose ASP.NET Web Application from the middle pane.

Step 4: Type “MVCControllerDemo” as the project name in the Name field, then click “OK” to proceed. The following dialog box will appear and ask you to specify the ASP.NET project’s initial content.

Step 5: To ensure simplicity, first choose the Empty option, then in the ‘Add directories and core references for’ section, check the MVC checkbox and click OK.

It will produce a simple MVC project with very little preset content.

The Solution Explorer window will show a variety of files and

folders once Visual Studio creates the project.

Step 6: In the solution explorer, right-click on the Controllers folder to add EmployeeController. Navigate to Add → Controller.

MVC Tutorial 18

The Add Scaffold dialog will appear.

MVC Tutorial 19

Step 7: Click the “Add” button after selecting the MVC 5 Controller-Empty option.

You will see the Add Controller dialog box.

Step 8: Click the “Add” button after renaming the object EmployeeController.

The Controllers folder will have a new C# file called EmployeeController.cs, which is editable in Visual Studio as well.

We will now implement a new route for the Employee controller along with the default Route in this application.

Step 1: Add the following route to the “RouteConfig.cs” file located in the “App_Start” folder.

```
routes.MapRoute(
  “Employee”, “Employee/{name}”, new{
    controller = “Employee”, action = “Search”, name =
    UrlParameter.Optional });
```

The RouteConfig.cs file’s full implementation is shown below.

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

namespace MVCControllerDemo {

  public class RouteConfig {

    public static void RegisterRoutes(RouteCollection
routes){

      routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

      routes.MapRoute(

        "Employee", "Employee/{name}", new{

          controller = "Employee", action = "Search",
name = UrlParameter.Optional });

      routes.MapRoute(

        name: "Default", url:
"{controller}/{action}/{id}", defaults: new{

          controller = "Home", action = "Index", id =
UrlParameter.Optional });

    }

  }

}
```

Step 2: Use the following code to modify the EmployeeController class.

```
using System;

using System.Collections.Generic;

using System.Linq;
```

```
using System.Web;
using System.Web.Mvc;
namespace MVCControllerDemo.Controllers  {
  public class EmployeeController : Controller {
    // GET: Employee
    public ActionResult Search(string name){
      var input = Server.HtmlEncode(name);
      return Content(input);
    }
  }
}
```

Output

MVC Model

A model is a class that holds the application's business logic.

Accessing data from the database is another purpose for it. The model class doesn't deal with browser input directly. Moreover, it is devoid of HTML code.

Objects used to implement conceptual logic for the application are also referred to as models. A controller communicates with the model, retrieves data, applies logic, and transfers data to the display.

Creating a Model

Let's update the project with a fresh model. For its characteristics, the model has setters and getters. Simply perform right-click on the project's Model folder and adhere to these instructions to add the model. *Visual C#->Code->Class; Model->Add->New Item*.

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace MvcApplicationDemo.Models

{

   public class ClassicalMusic

   {

   }

}
```

The model can now have an infinite number of properties and methods added to it. These aid in creating a clean framework for MVC. Just like this, we're developing a method and a few properties.

```
using System;

using System.Collections.Generic;
```

```
using System.Linq;

using System.Web;

namespace MvcApplicationDemo.Models

{

   public class ClassicalMusic

   {

      public int ID { get; set; }

      public string Title { get; set; }

      public DateTime ReleaseDate { get; set; }

      public string Genre { get; set; }

      public string GetDateTime()

      {

         return DateTime.Now.ToString();

      }

   }

}
```

MVC View

An HTML page that is standard and may have a script on it is the MVC View. The application's web pages are made with it. MVC Views, in contrast to ASP.NET Web Pages, are mapped to the action, and the controller then renders the view for the browser.

Certain conventions for project structure are part of MVC. The subdirectory of the View folder is where you should find the view file.

Because MVC makes use of the Razor view engine, HTML may also be used to construct server-side code. Now let's create a view and run it in a browser.

Creating a View of the Application

Right-click on the subfolder within the View folder and choose Add -> Add View to add a view. For example, it will appear for the view name.

Welcome.cshtml

```
@{
    ViewBag.Title = "Welcome";
}
<h2>Welcome</h2>
```

To carry it out, we have a controller that looks like this:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace MvcApplicationDemo.Controllers
{
    public class StudentsController : Controller
    {
        // GET: Students
        public ActionResult Index()
        {
            return View();
        }
```

```
        public ActionResult Welcome()
        {
            return View();
        }
    }
}
```

The browser will see the Welcome view file thanks to this controller's Welcome action function. The Welcome.cshtml file can be seen in a browser by performing a right-click. Explore what our **advanced DotNet course** has in store for your career growth.

MVC-Related Frameworks

Several MVC-based frameworks are as follows:

- **ASP.NET Core MVC:** A lightweight, highly testable, open-source framework that is tailored for ASP.NET Core usage. It's a method of creating dynamic websites based on patterns.
- **Django:** A Python web application framework that is free and open-source. It offers a collection of parts for creating intricate, database-driven websites.
- **Laravel:** An intuitive PHP framework that serves as a foundation for creating small- to large-scale enterprise applications.
- **Ruby on Rails:** An MVC framework for creating server-side Ruby applications is called Ruby on Rails. For data transfers, it employs XML and JSON; for interfaces, it uses HTML/CSS and JavaScript.
- **Spring MVC:** An MVC architecture with components for creating web applications is called Spring MVC. DispatcherServlet, a servlet that routes and intercepts HTTP requests, is the foundation of this system.

Advantages of the ASP.NET MVC Framework

- By splitting a program into its model, view, and controller, it controls the complexity of the application.
- Neither server-based forms nor view states are used. Because of this, developers who desire complete control over an application's behavior will find the MVC framework to be perfect.
- Better assistance for test-driven development is offered by

it.

- Web applications and large-scale development teams can benefit from it.
- It gives the developer a great deal of control over how the application behaves.

Conclusion

We have covered the fundamentals in this MVC tutorial. Train your brain with hands-on exposure through our **MVC training in Chennai**.

Share on your Social Media

Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318
Email: enquiry@softlogicsys.in
Map: Google Maps Link

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.
Landmark: Adj. to AGS Cinemas
Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

- Full Stack Developer
- Java
- Power BI
- Clinical SAS
- Data Science
- Embedded
- Cloud Computing
- Hardware and Networking
- VBA Macros
- Mobile App Development
- DevOps

Review Sources

- Google
- Trustpilot
- Glassdoor
- Mouthshut
- Sulekha
- Justdial
- Ambitionbox
- Indeed
- Software Suggest
- Sitejabber