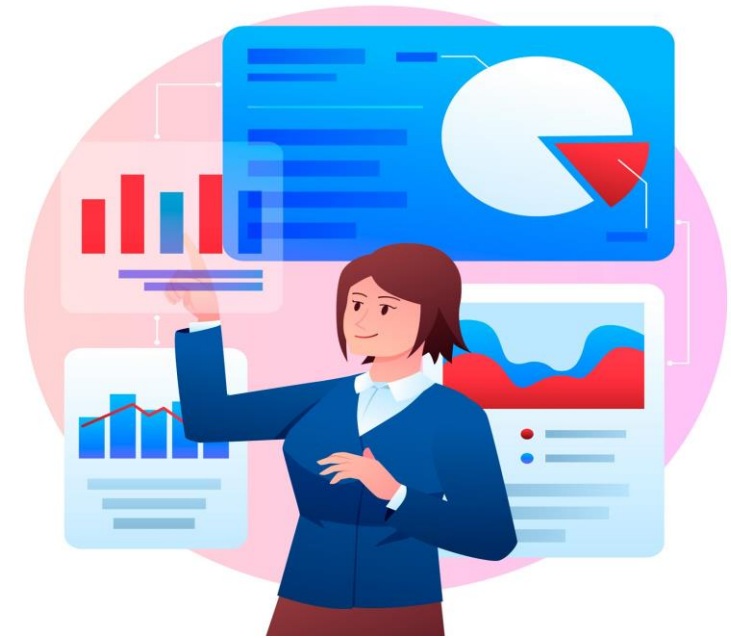


MVC (Model-View- Controller)



What is MVC?

- The MVC (Model-View-Controller) design pattern was introduced in the 1970s which divides an application into 3 major components. They are Model, View, and Controller.
- MVC stands for Model View and Controller.
- It is an architectural design pattern that means this design pattern is used at the architecture level of an application.
- MVC is not a Framework, it is a design pattern.



What is ASP.NET Core MVC?

- The ASP.NET Core MVC is a lightweight, open-source, highly testable presentation framework that is used for building web apps and Web APIs using the Model-View-Controller (MVC) design pattern.
- MVC is a design pattern and ASP.NET Core MVC is the framework that is based on MVC Design Pattern.
- It also supports for Test-Driven Development and also uses the latest web standards.

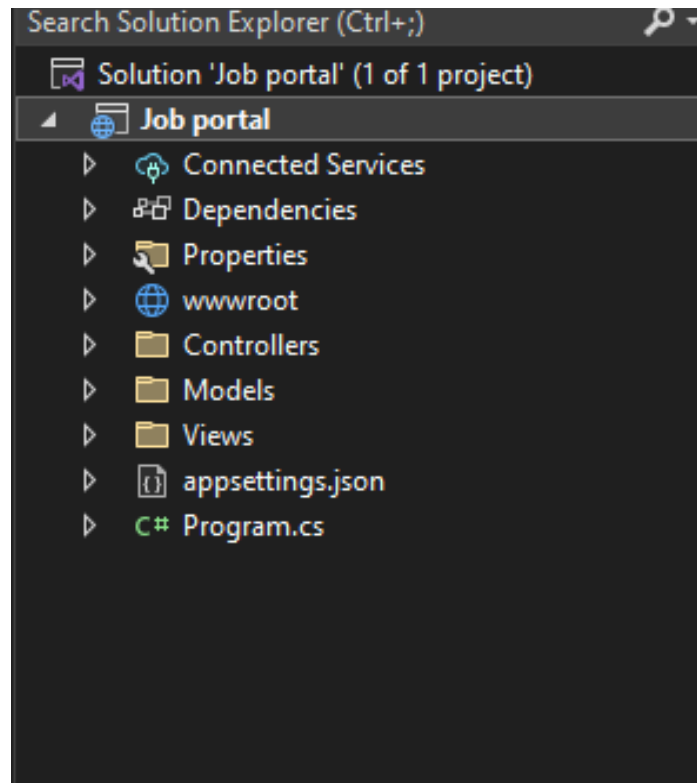


How to Set up MVC in ASP.NET Core

Create New Project

Configure your new project

Addition information



01

Model: A set of classes that describes the data you are working with as well as the business logic.



02

Controller: A set of classes that handles communication from the user, overall application flow, and application-specific logic.

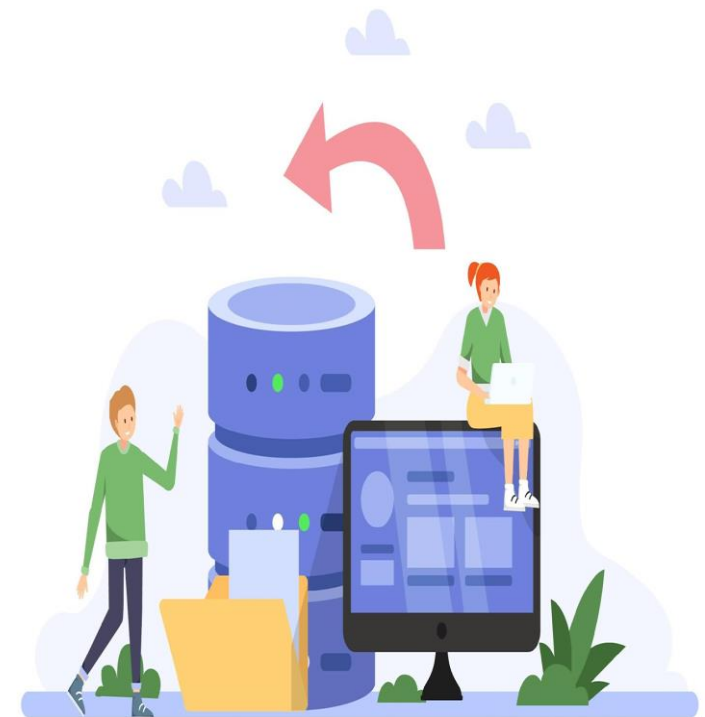


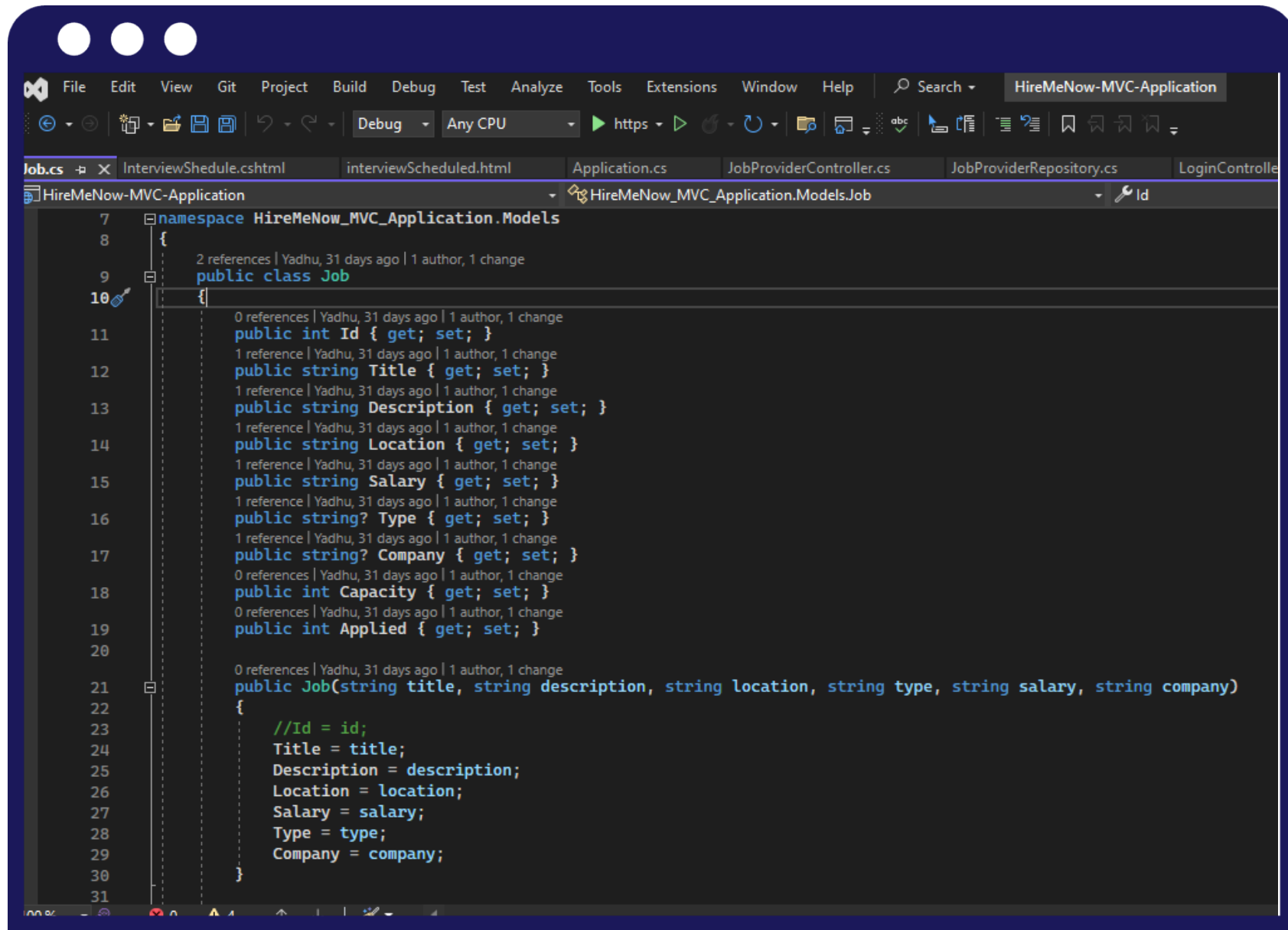
03

View: Defines how the application's UI will be displayed. It is a pure HTML, which decides how the UI is going to look like.

Model

- A model is a class with .cs (for C#) as an extension having both properties and methods.
- Models are used to set or get the data.
- If your application does not have data, then there is no need for a model.
- models in ASP.NET Core MVC Application are used to manage the data i.e. the state of the application in memory.





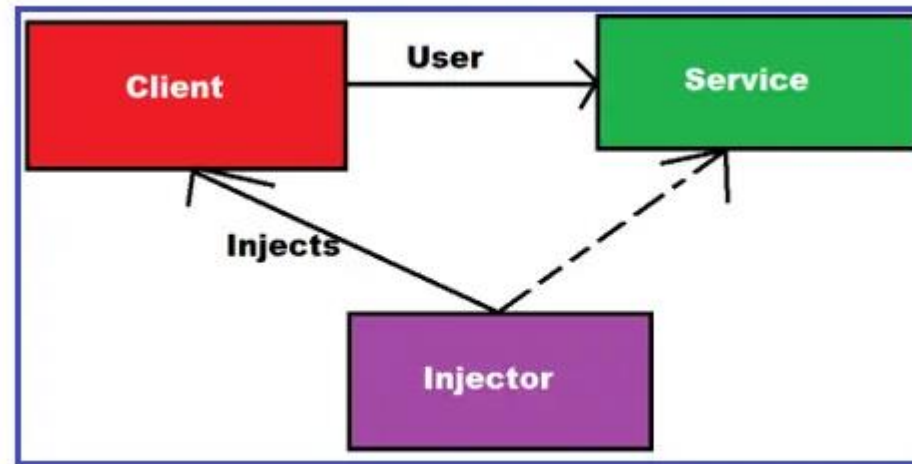
The screenshot shows the Visual Studio IDE with the 'HireMeNow-MVC-Application' project open. The 'Job.cs' file is selected, and the 'HireMeNow_MVC_Application.Models.Job' namespace is expanded. The code defines a 'Job' class with the following properties and methods:

```
7 namespace HireMeNow_MVC_Application.Models
8 {
9     public class Job
10    {
11        public int Id { get; set; }
12        public string Title { get; set; }
13        public string Description { get; set; }
14        public string Location { get; set; }
15        public string Salary { get; set; }
16        public string? Type { get; set; }
17        public string? Company { get; set; }
18        public int Capacity { get; set; }
19        public int Applied { get; set; }
20
21        public Job(string title, string description, string location, string type, string salary, string company)
22        {
23            //Id = id;
24            Title = title;
25            Description = description;
26            Location = location;
27            Salary = salary;
28            Type = type;
29            Company = company;
30        }
31    }
```

ASP.NET Core Dependency Injection

process of injecting the object of a class into a class that depends on it.

The Dependency Injection is the most commonly used design pattern nowadays to the dependencies between the objects that allow us to develop loosely coupled software components.



The built-in container is represented by `IServiceProvider` implementation that supports constructor injection by default. The types (classes) managed by built-in IoC containers are called services.

Types of Services in ASP.NET Core:

Framework Services: Services that are a part of the ASP.NET Core framework such as `IApplicationBuilder`, `IHostingEnvironment`, `ILoggerFactory`, etc.

Application Services: The services (custom types or classes) which you as a programmer create for your application.

In order to let the IoC container automatically inject our application services, we first need to register them with the IoC container

How to register a Service with ASP.NET Core Dependency Injection Container?

- We need to register a service with ASP.NET Core Dependency Injection Container within the `ConfigureServices()` method of the Startup class.

Singleton: In this case, the IoC container will create and share a single instance of a service object throughout the application's lifetime.

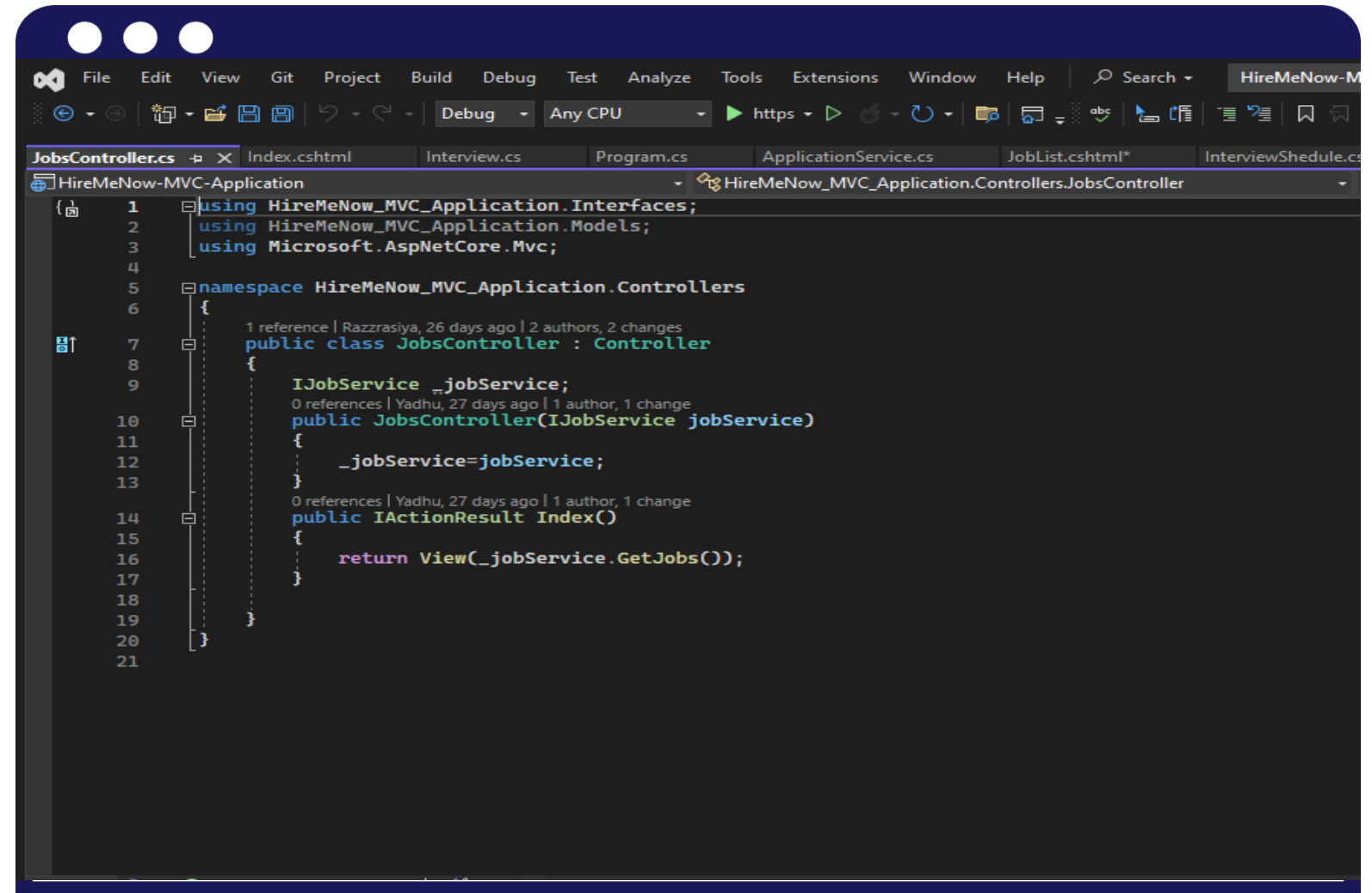
Transient: In this case, the IoC container will create a new instance of the specified service type every time you ask for it.

Scoped: In this case, the IoC container will create an instance of the specified service type once per request and will be shared in a single request.

Controller

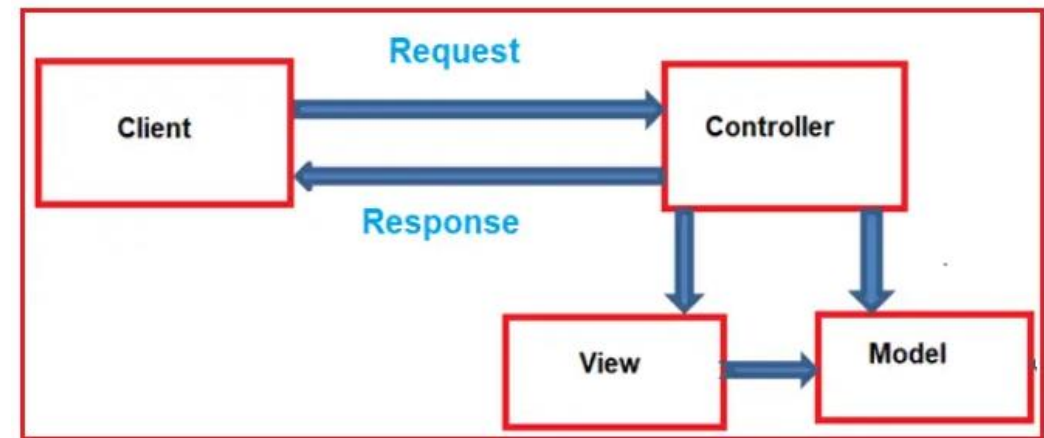
A Controller is a .cs (for C# language) file which has some methods called Action Methods.

When a request comes on the controller, it is the action method of the controller which will handle those requests.



```
1 using HireMeNow_MVC_Application.Interfaces;
2 using HireMeNow_MVC_Application.Models;
3 using Microsoft.AspNetCore.Mvc;
4
5 namespace HireMeNow_MVC_Application.Controllers
6 {
7     1 reference | Razzrasiya, 26 days ago | 2 authors, 2 changes
8     public class JobsController : Controller
9     {
10         IJobService _jobService;
11         0 references | Yadhu, 27 days ago | 1 author, 1 change
12         public JobsController(IJobService jobService)
13         {
14             _jobService=jobService;
15         }
16         0 references | Yadhu, 27 days ago | 1 author, 1 change
17         public IActionResult Index()
18         {
19             return View(_jobService.GetJobs());
20         }
21     }
```

When the client (browser) sends a request to the server, then that request first goes through the request processing pipeline. Once the request passes the request processing pipeline, it will hit the controller. Inside the controller, there are lots of methods (called action methods) actually handle that incoming HTTP Request. The action method inside the controller executes the business logic and prepared the response which is sent back to the client who initially made the request.



What are Action Methods?

All the public methods of a controller class are known as Action Methods.

An action method can return several types.

When we get an HTTP Request on a Controller, it is actually the controller action method getting that call.
The default structure is:
`http:domain.com/ControllerName/ActionMethodName`

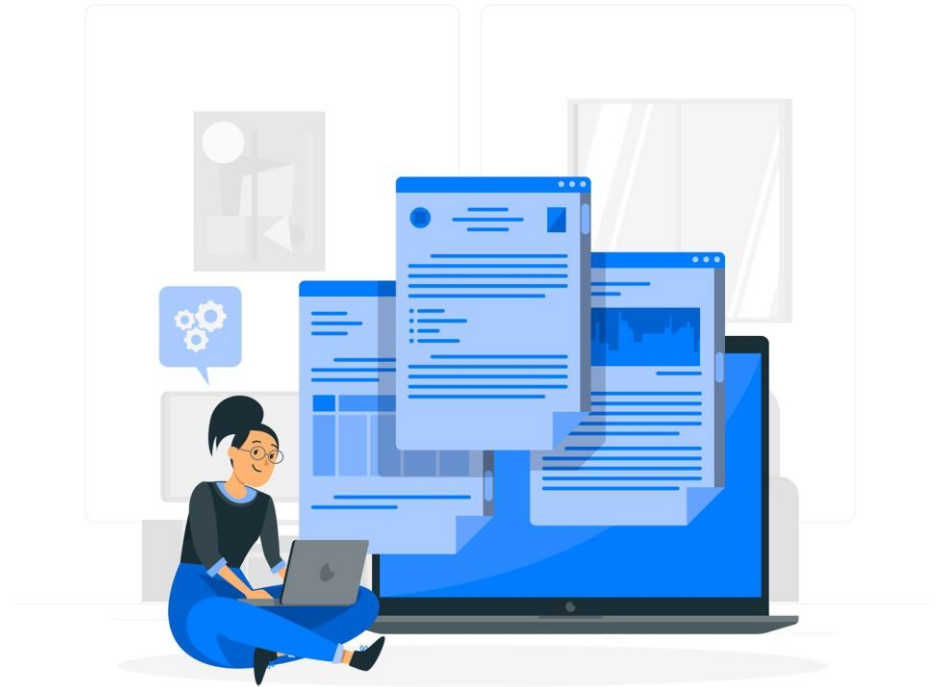


Views in ASP.NET Core MVC

View is a file with “.cshtml” (for C# language) extension.

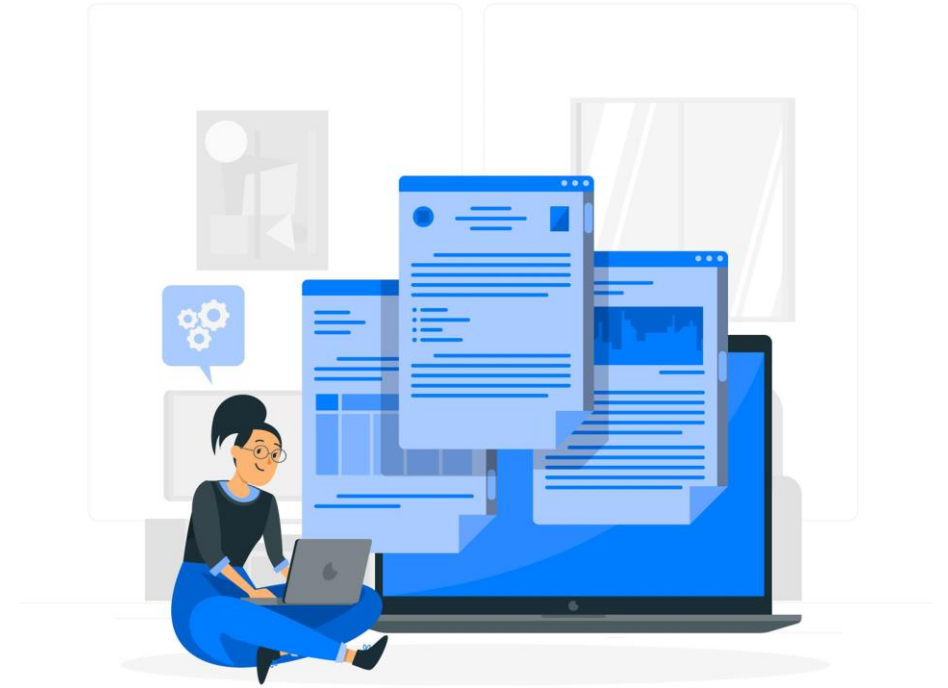
In the Model-View-Controller (MVC) pattern, the View is the component that contains logic to represent the model data (the model data provided to it by a controller) as a user interface with which the end-user can

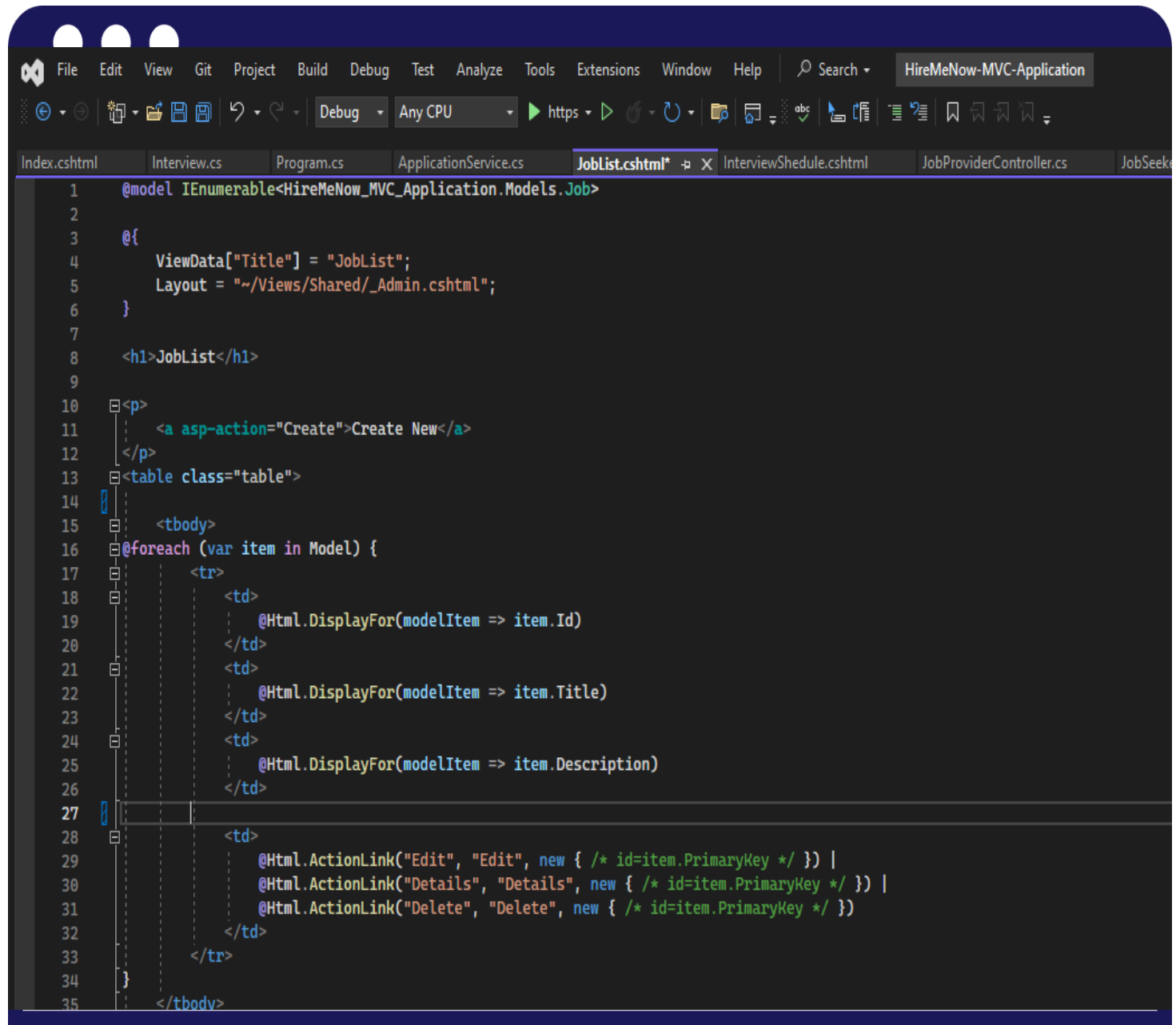
The Views in MVC are HTML templates with embedded Razor mark-up which generate content that sends to the client.



Views are returned from the controller
Action Method

View file name is the same as the action
method name of a controller with the
“.cshtml” extension.





```
1  @model IEnumerable<HireMeNow_MVC_Application.Models.Job>
2
3  @{
4      ViewData["Title"] = "JobList";
5      Layout = "~/Views/Shared/_Admin.cshtml";
6  }
7
8  <h1>JobList</h1>
9
10 <p>
11     <a asp-action="Create">Create New</a>
12 </p>
13 <table class="table">
14
15     <tbody>
16     @foreach (var item in Model) {
17         <tr>
18             <td>
19                 @Html.DisplayFor(modelItem => item.Id)
20             </td>
21             <td>
22                 @Html.DisplayFor(modelItem => item.Title)
23             </td>
24             <td>
25                 @Html.DisplayFor(modelItem => item.Description)
26             </td>
27
28             <td>
29                 @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) |
30                 @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |
31                 @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })
32             </td>
33         </tr>
34     }
35 </tbody>
```


How does MVC Design Pattern work in ASP.NET Core?

1. Controller Handles the incoming request.
2. Controller component creates the model that is required by a view .
3. Once the model created by the controller, then the controller selects a view to render the domain data or model data.
4. While selecting a view, it is also the responsibility of the controller to pass the model data.
5. View generates the necessary HTML render the model data.
6. Once the HTML is generated the view is send to the client through the network

