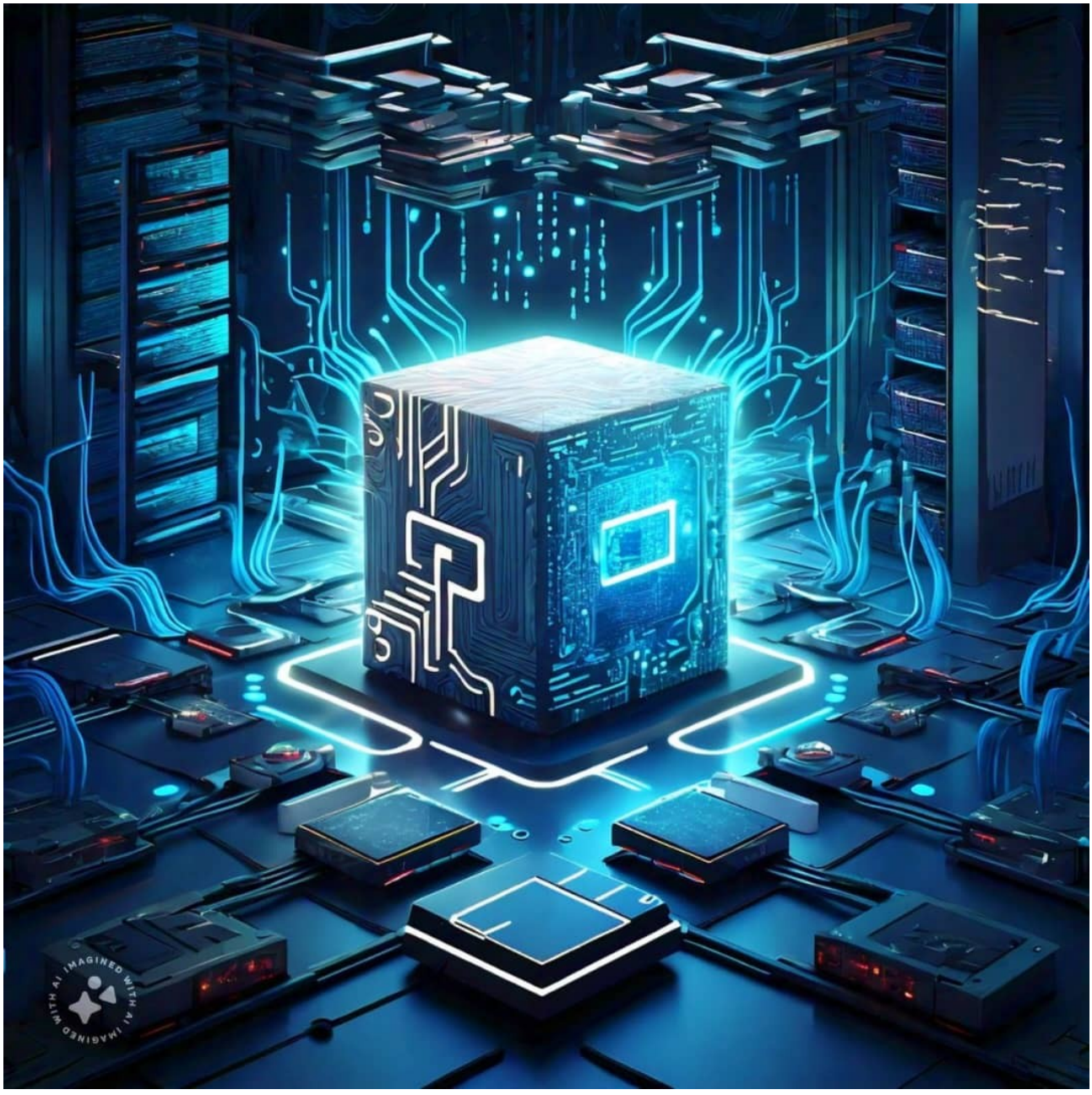
Hydot Backend Tutorial



Master 5 Popular Backend Frameworks Step By Step

(1) Dotnet mvc and Api
(2) Laravel mvc and Api
(3) Spring boot mvc and Api
(4) Django mvc and Api
(5) Express Node Api

DOTNET MVC

[Start]

From the terminal run this command **dotnet new mvc -o ProjectName**
Open the solution in VSCode **code ProjectName**

When Visual Studio Code requests that you add assets to build and debug the project, select **Yes**. If Visual Studio Code doesn't offer to add build and debug assets, select **View** > **Command Palette** and type ".NET" into the search box. From the list of commands, select the .NET: Generate Assets for Build and Debug command.

Visual Studio Code adds a .vscode folder with generated launch.json and tasks.json file

Trust the HTTPS development certificate by running the following command:
dotnet dev-certs https --trust

In vscode, press ctrl + f5 to run the app without debugging

[Application]

The typical flow of the application is Model View Controller(MVC) and each of them play a critical role
Model: It defines the data schema that will be used to write data into the database.
View: It defines the UI of the application
Controller: It correlate the Model and View Activity. It stores the application logic and process the users request, it then response to the user request with a view.

Ui logic belongs to the View, Input logic belongs to the Controller and the business logic belongs to the model

Create a new file name it HelloWorld.cs and paste this content into it

```
using Microsoft.AspNetCore.Mvc;
using System.Text.Encodings.Web;

namespace MvcMovie.Controllers;

public class HelloWorldController: Controller{

public string Index(){

return "Hello World";
}

public string Welcome(){
return "Welcome";
}



}
```

Every public method is a Http endpoint.

The url is https://localhost:7111/HelloWorld
https => The protocol used
localhost:7111 => The server and port running the application
HelloWorld => The controller Name, [HelloWorldController], inside this controller, we have the Index() method. This index method will be the initial method which will be executed when the user access the HelloWorld controller
The Index() is the base url.

https://localhost:7111/HelloWorld/Welcome
Now for this example, we are moving into the welcome method Welcome() to execute the code there. The controller name is HelloWorld and the specific method name is Welcome

/[Controller]/[ActionName]/[Parameters]

In the program.cs, the default route is

```
app.MapControllerRoute(
name: "default",
pattern: "{controller=Home}/{action=Index}/{id?}"
);
```

We can modify it so that the default route will be HelloWorldController

```
app.MapControllerRoute(
name: "default",
pattern: "{controller=HelloWorld}/{action=Index}/{id?}"
);
```

Parameters
From the welcome method in the HelloWorld controller add this

```
public string Welcome(string name, int numTimes = 1)
{
return HtmlEncoder.Default.Encode($"Hello {name}, NumTimes is: {numTimes}");
}
```

In your browser do this https://localhost:7111/HelloWorld/Welcome?name=Rick&numtimes=4

Add A View

In the View folder, create a folder called HelloWorld. Note that the folder name in the View folder should match with the controller name.

If the controller name is MoneyController, the folder name should be Money and so on

In the HelloWorld folder, create an Index.cshtml which is a razor view file,

The code should be this

```
@{
ViewData["Title"] = "Index";
}
```

```
<h2>This is the index file</h2>

<p>Hello from the view template!</p>
```

Now in the HelloWorldController, update the Index()

```
public IActionResult Index(){

return View();
}
```

Now this code will do this, It will move into the View Folder > HelloWorld Folder > Index.cshtml

Layout

In the shared folder, there is a _Layout.cshtml file that will act as the root of the application. Now in this file, the section is divided into 3 main section

<header> </header>

<main> </main>

<footer> </footer>

The header contains the normal header of the application, it can be modified

```
<header>
<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white
border-bottom box-shadow mb-3">
<div class="container-fluid">
<a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">MvcMovie</a>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target=".navbar-collapse" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
<ul class="navbar-nav flex-grow-1">
<li class="nav-item">
<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="Index">Home</a>
</li>
<li class="nav-item">
<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="Privacy">Privacy</a>
</li>
</ul>
</div>
</div>
</nav>
```

```
</header>
```

The main section will display all the route i will enter

```
<div class="container">
<main role="main" class="pb-3">
@RenderBody()
</main>
</div>
```

@RenderBody() => RenderBody is a placeholder where all the view-specific pages you create show up, *wrapped* in the layout page. For example, if you select the Privacy link, the *Views/HelloWorld/Index.cshtml* view is rendered inside the RenderBody method.

The Footer section will contain the normal footer.

```
<footer class="border-top footer text-muted">
<div class="container">
&copy; 2024 - MvcMovie - <a asp-area="" asp-controller="Home" asp-
action="Privacy">Privacy</a>
</div>
</footer>
```

Good, now the scripts resides in the wwwroot folder. The [~] sign means wwwroot folder

```
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>
```

Okay, the Views/_ViewStart.cshtml contain this code

```
@{
Layout = "_Layout";
}
```

To display the title of the page, modify the Index.cshtml

```
@{
ViewData["Title"] = "Movie List";
}


<title>@ViewData["Title"]-Movie App</title>

<h2>This is the index file</h2>

<p>Hello from the view template!</p>
```