**CSPROJ-Project file**

Have a project file end with .csproj. It contains information about the target framework (the version of .NET) and package references ,nullable and implicitusing

**Hosting model**

How the application is hosted (in-process or out-of-process).

In-process: The program is hosted inside IIS Express.

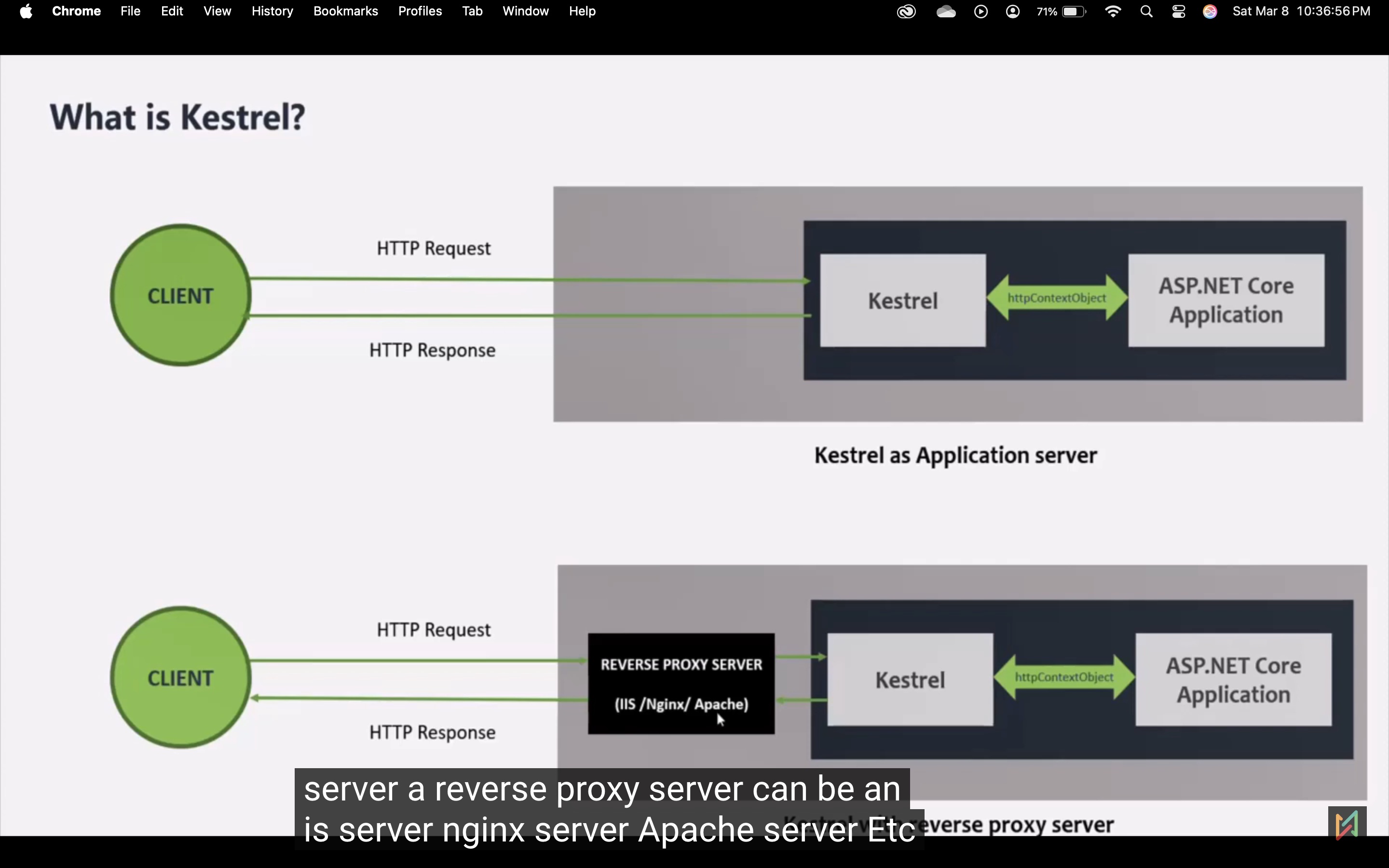
Out-of-process: The program is hosted internal and webserver.

Internal🡪kerstel(crsoo-platform webserver)

External🡪IIS,nginx,apache

Kestrel🡪cross platform webserver for .netcore Apps,iis is only for windows

It can also be a application server, reverse proxy server



In the diagram above, we’re discussing the development process. During development, we receive requests from the webpage. The Kestrel receives these requests, creates an HTTP context object, and sends it to the .NET Core application. After the application processes the request, the Kestrel receives the HTTP context object and sends a response back to the browser.

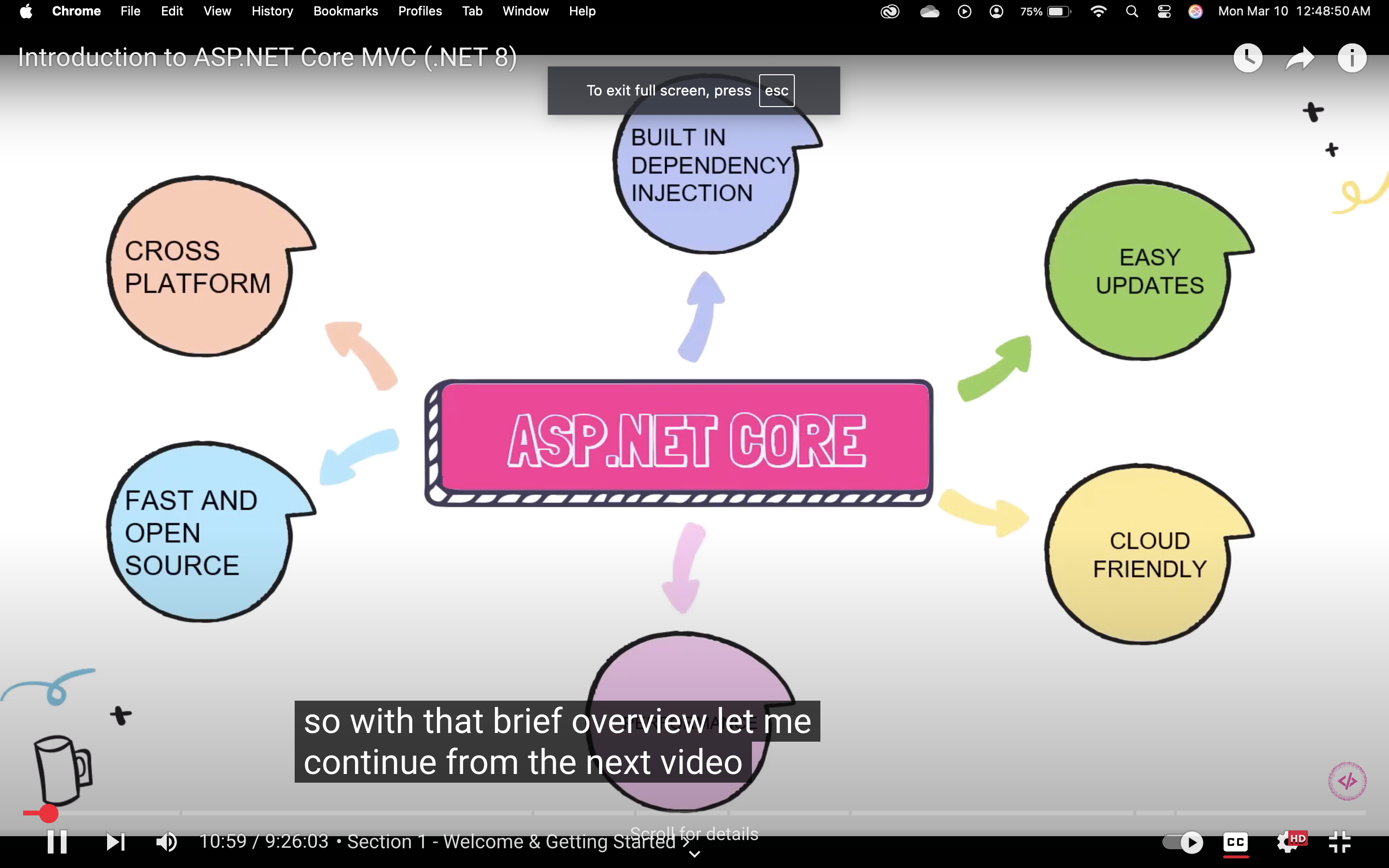
In a production environment, kestrel alone is not sufficient to handle browser requests. To address this, we also use a proxy server.

Once the “.NET Core Web Application” run (HTTP, hotload) button is clicked, a console application will be started. This application will then start the Kestrel server, where the application will be hosted.

**client server architecture**

<https://www.youtube.com/watch?v=KSPlGDxOilM>

<https://www.youtube.com/watch?v=a3LHYguyHkw&list=PL1BztTYDF-QPgfvPouABKLwfTKxB6z7gk&index=14>



**Lauchsettings.json**

Which profile should we use for the application? The profiles are http, https, and iisexpress. The URL details and environment details will be available.

**Wwwrott**

Static files, such as CSS and JavaScript, will be placed here.

**Appsettings.json**

Connection string, secret key

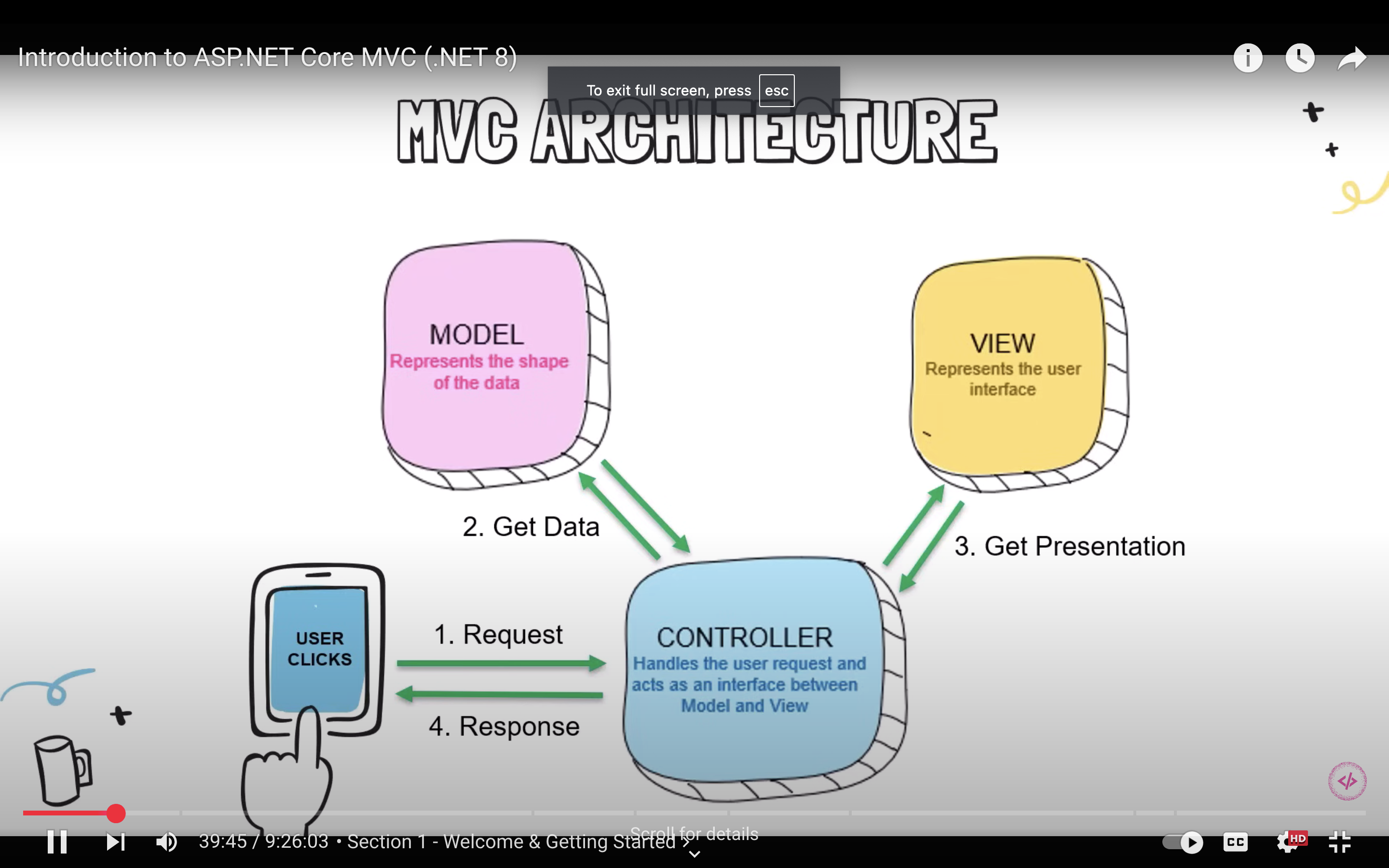
**Program.cs**

Configure a service, create a middleware

Model🡪Represents the shape of the data

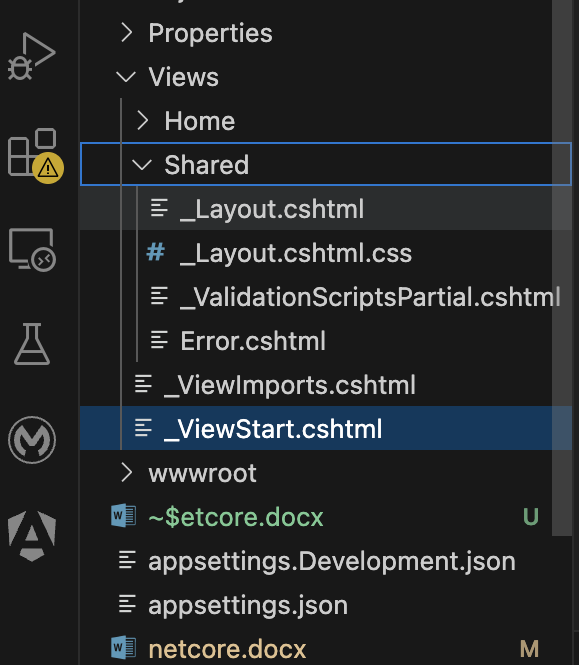
View🡪represent the user Interface

Controller🡪Handles the request and act as an interface between between model and view



Once the user clicks on the browser, the request goes to the controller. The controller retrieves data from the model and passes it back to the controller. The controller calls the view, which creates an HTML page representation based on the data. The view then passes the representation back to the controller. Finally, the controller sends back the response.

**Views**



\_Layout.cshtml: The master page contains the header, footer, and uses @RendorBody() to render the body content.

\_validationscriptpartial: This partial view adds client-side validation.

Error.cstml: This view handles error messages.

\_ViewStart.cshtml: This view specifies which page is using the layout.

\_ViewImport.cshtml:Instead of importing each time in a different file, it uses a global import, eliminating the need to import in each file. **only available in views**

**Controller**

Return IACTIONRESULT