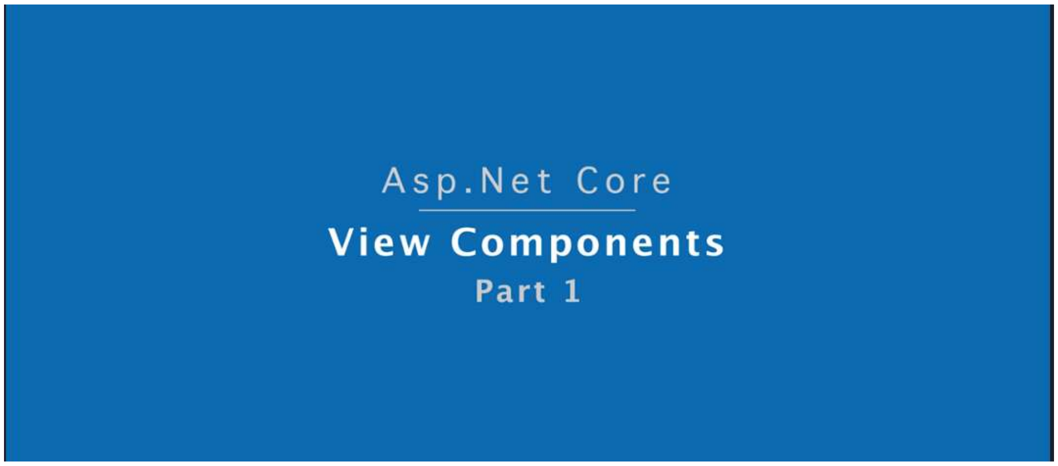
Section 11: View Components [MVC]

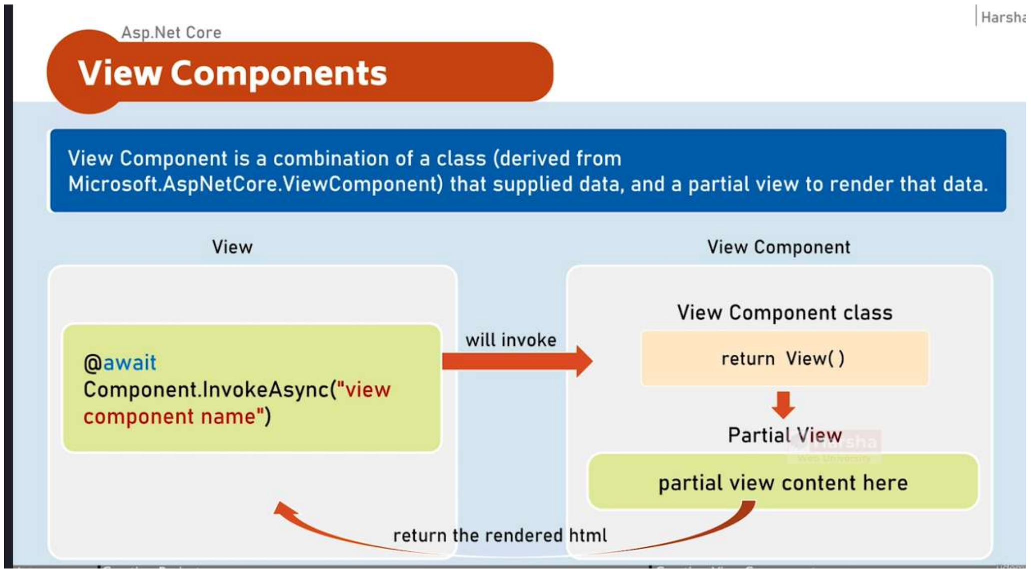
Thursday, January 16, 2025 6:45 PM



Asp.Net Core

View Components

Part 1



Asp.Net Core

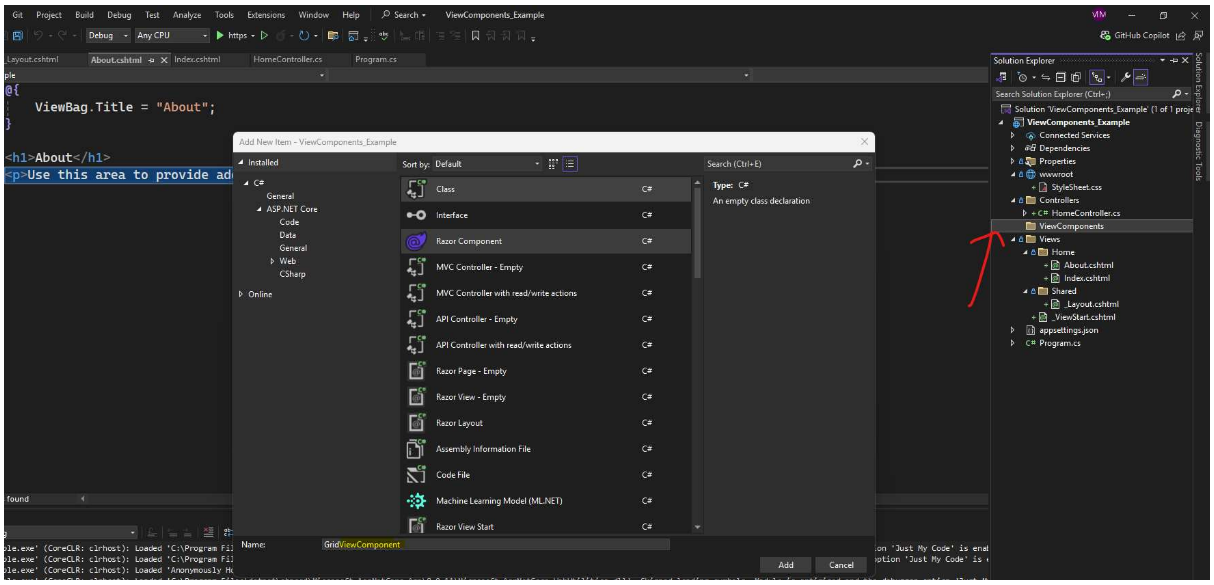
View Components

View Component is a combination of a class (derived from Microsoft.AspNetCore.ViewComponent) that supplied data, and a partial view to render that data.

The partial view is primarily used for rendering the UI. It allows you to place reusable HTML code, i.e., presentation logic, in a partial view, which can be invoked anytime.
However, when you need to combine additional programming logic with UI logic, a **view component** is the preferred choice. For example, you can create a **view component class** to include programming logic, such as performing calculations, retrieving data from a database, or preparing model data to supply to the view. The partial view, which is part of the view component, will then handle the UI rendering.
In general, when you want to integrate programming logic with UI logic as a single unit, a **view component** is the ideal choice. Just like a partial view, a view component can be invoked in any other view as needed by using the following statement:

@Component.InvokeAsync("ComponentName")

Let me demonstrate this practically.



```
@{
    ViewBag.Title = "About";
}

<h1>About</h1>
<p>Use this area to provide ad
```

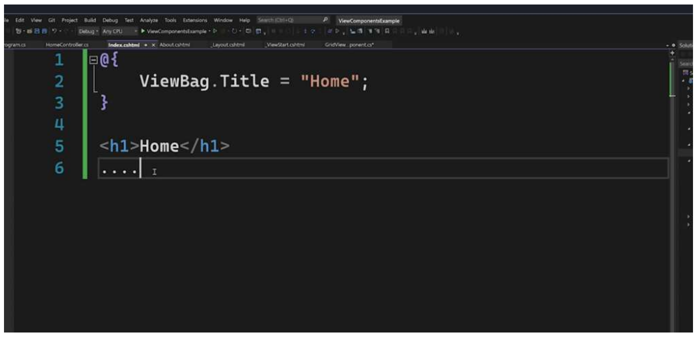
Class	C#
Interface	C#
Razor Component	C#
MVC Controller - Empty	C#
MVC Controller with read/write actions	C#
API Controller - Empty	C#
API Controller with read/write actions	C#
Razor Page - Empty	C#
Razor View - Empty	C#
Razor Layout	C#
Assembly Information File	C#
Code File	C#
Machine Learning Model (ML.NET)	C#
Razor View Start	C#

Type: C#
An empty class declaration

Name: GridViewComponent

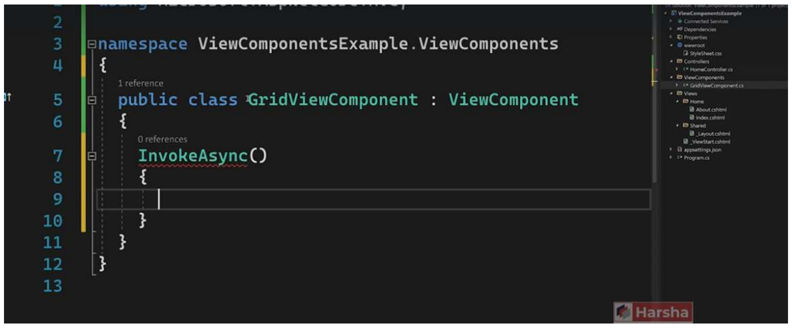
```
Solution 'ViewComponents_Example' (1 of 1 proje
  ViewComponents_Example
    Connected Services
    Dependencies
    Properties
    wwwroot
      StyleSheet.css
    Controllers
      HomeController.cs
    ViewComponents
    Views
      Home
        About.cshtml
        Index.cshtml
      Shared
        _Layout.cshtml
        _ViewStart.cshtml
    appsettings.json
    Program.cs
```

Suffix the word, 'ViewComponent'

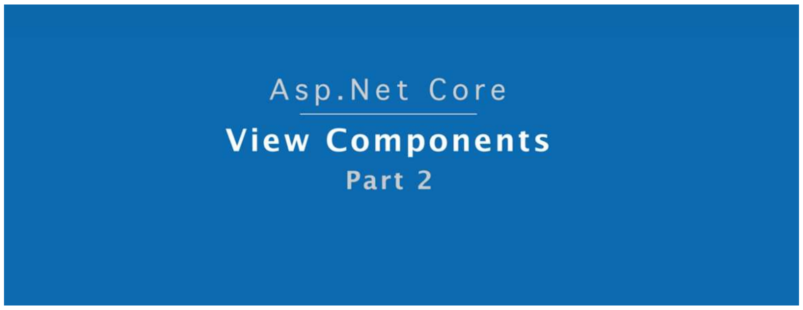


```
1  @{
2      ViewBag.Title = "Home";
3  }
4
5  <h1>Home</h1>
6  ....
```

From Index View, lets assume that at line no 6, we are calling ViewComponent. Then, it automatically creates an object of 'GridViewComponent' and it calls 'InvokeAsync()' method.



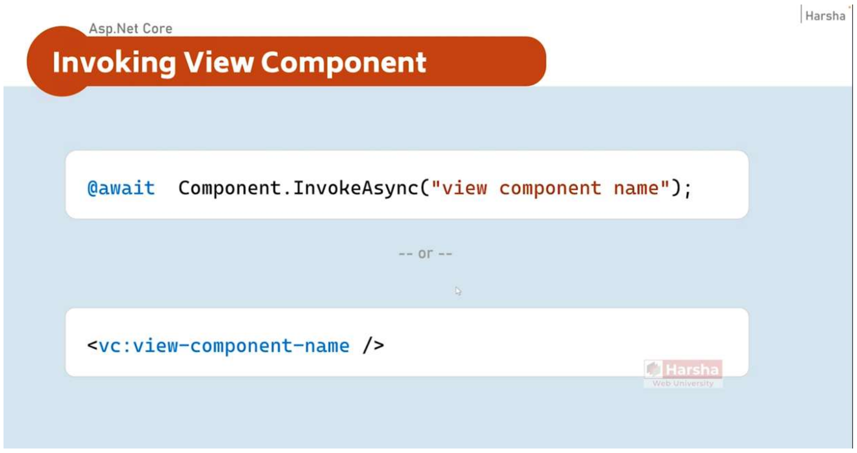
```
namespace ViewComponentsExample.ViewComponents
{
    public class GridViewComponent : ViewComponent
    {
        InvokeAsync()
        {

        }
    }
}
```



Asp.Net Core

View Components

Part 2



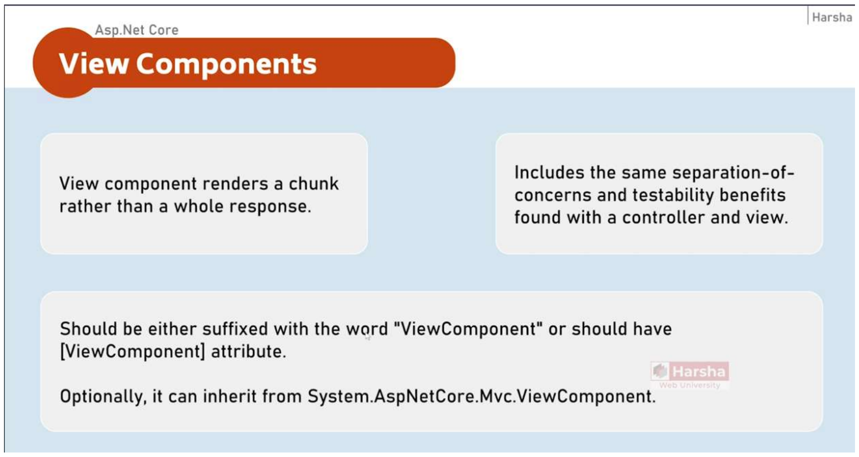
Asp.Net Core

Invoking View Component

```
@await  Component.InvokeAsync("view component name");
```

-- or --

```
<vc:view-component-name />
```



Asp.Net Core

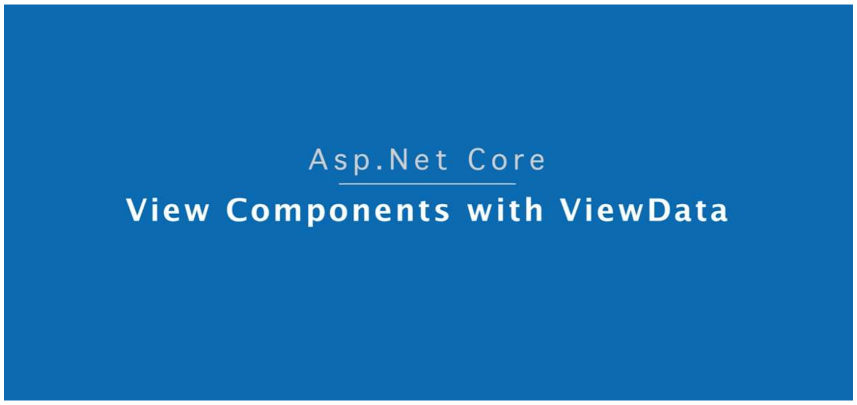
View Components

View component renders a chunk rather than a whole response.

Includes the same separation-of-concerns and testability benefits found with a controller and view.

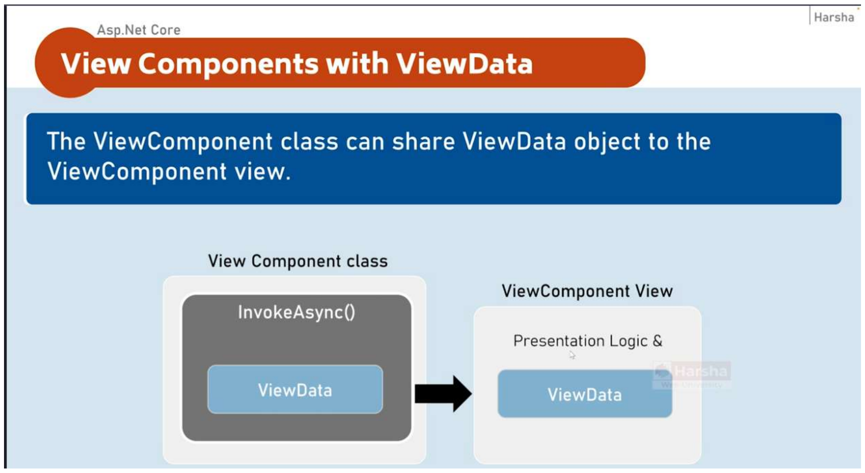
Should be either suffixed with the word "ViewComponent" or should have [ViewComponent] attribute.

Optionally, it can inherit from System.AspNetCore.Mvc.ViewComponent.



Asp.Net Core

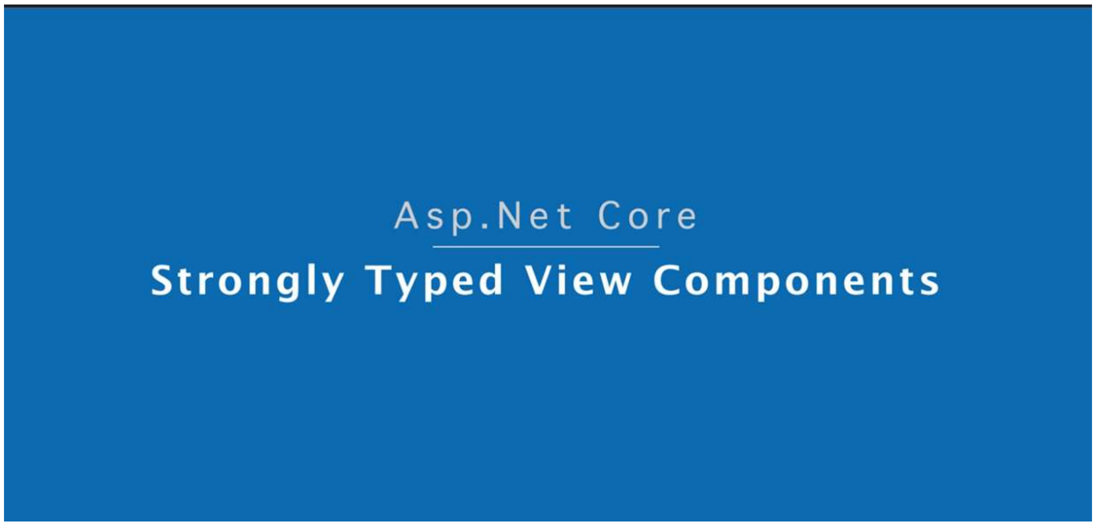
View Components with ViewData



View Components with ViewData

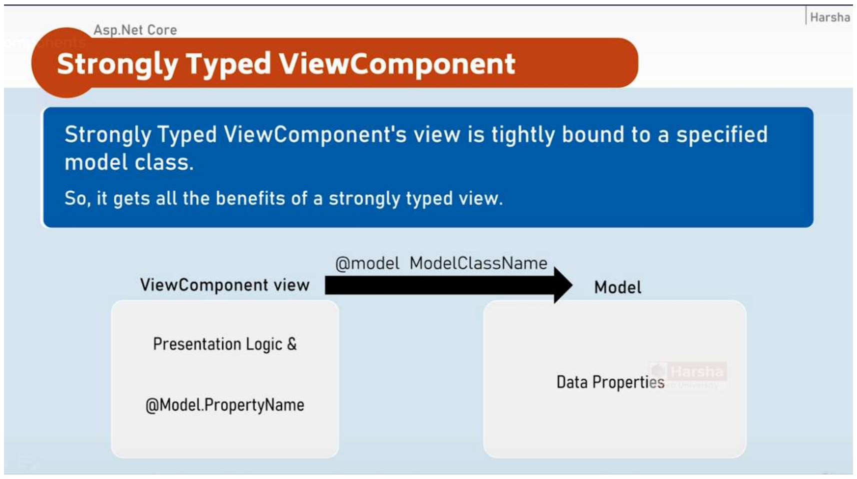
The ViewComponent class can share ViewData object to the ViewComponent view.

It is possible to supply the **ViewData** object from the **view component class** to the **view component partial view**. However, it is **not possible** to supply the **ViewData** object from the **view** to the **view component class**.
In this lecture, we will focus on sharing the **ViewData** object from the **view component class** to the **view component partial view**, similar to how you share the **ViewData** from the **controller** to the **view**.
For example, if you want to send a model object from the **view component class** to the **partial view**, you can achieve this by passing the model directly as part of the view component logic.
Let me show how this can be done practically.



Asp.Net Core

Strongly Typed View Components


Asp.Net Core

Harsha

Strongly Typed ViewComponent

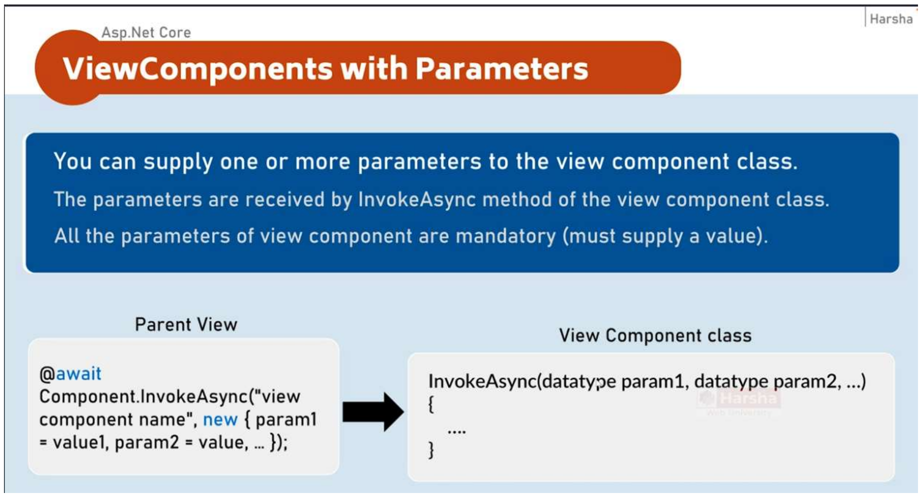
Strongly Typed ViewComponent's view is tightly bound to a specified model class.

So, it gets all the benefits of a strongly typed view.

Just like you can make a **view** or **partial view** a **strongly typed view** bound to a specific model class, you can also make a **view component** a **strongly typed view component**.
This means the **partial view** of the **view component** will be strongly typed to a specific model class. The **view component class** can then supply a model object to the corresponding **partial view**.
The **partial view** of the **view component** receives the model object and renders it in the UI. This approach provides all the advantages of a strongly typed view and ensures seamless interaction between the view component class and its partial view.

Asp.Net Core

View Components with Parameters


Asp.Net Core

Harsha

ViewComponents with Parameters

You can supply one or more parameters to the view component class.

The parameters are received by InvokeAsync method of the view component class.

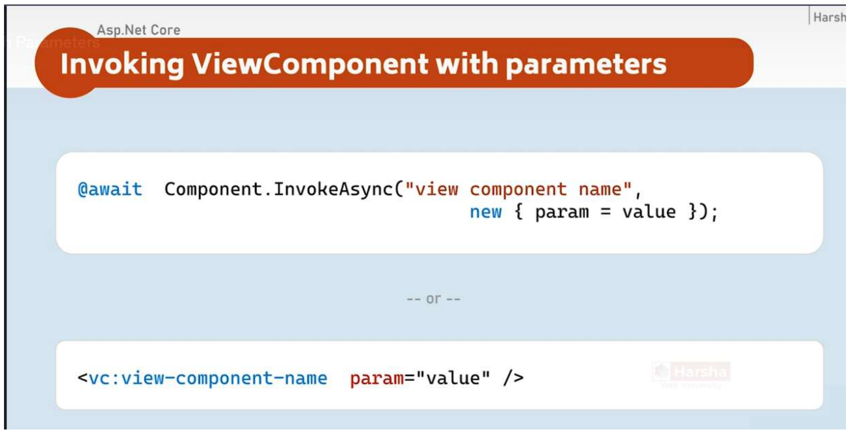
All the parameters of view component are mandatory (must supply a value).

Parent View

```
@await
Component.InvokeAsync("view
component name", new { param1
= value1, param2 = value, … });
```

View Component class

```
InvokeAsync(datatype param1, datatype param2, ...)
{
  ....
}
```



Asp.Net Core

Invoking ViewComponent with parameters

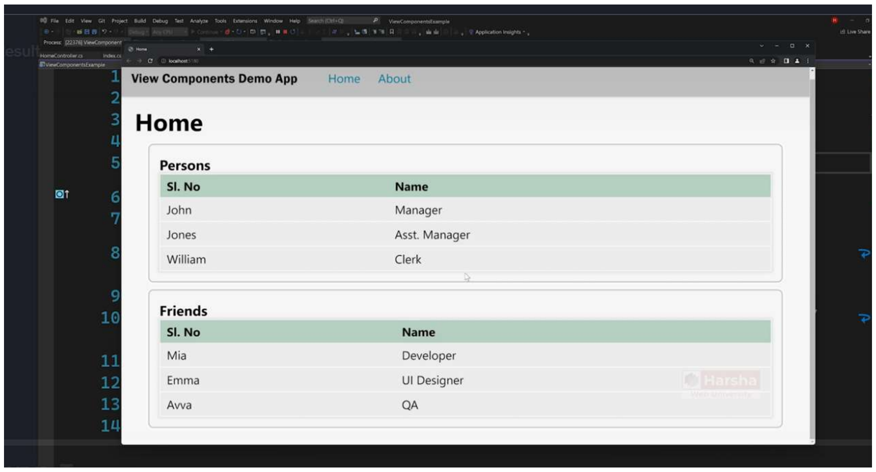
```
@await  Component.InvokeAsync("view component name",
                              new { param = value });
```

-- or --

```
<vc:view-component-name  param="value" />
```

Asp.Net Core

ViewComponentResult



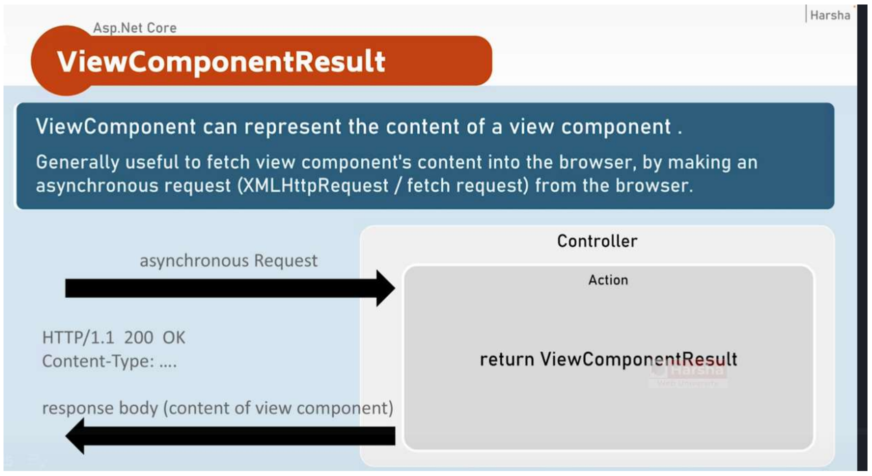
Home

Persons

Sl. No	Name
John	Manager
Jones	Asst. Manager
William	Clerk

Friends

Sl. No	Name
Mia	Developer
Emma	UI Designer
Avva	QA



Asp.Net Core

ViewComponentResult

ViewComponent can represent the content of a view component .

Generally useful to fetch view component's content into the browser, by making an asynchronous request (XMLHttpRequest / fetch request) from the browser.

In the current application, the **view component** is invoked automatically at runtime as soon as the view loads. However, there might be scenarios where you want to invoke the **view component programmatically** through JavaScript by making an asynchronous request.
For instance, you may want the view to load **without the view component initially**, and invoke the **view component** only **on demand**—for example, when a user clicks a button.
This is achievable by returning a **ViewComponentResult** from a **controller action method**. This way, the view component is loaded dynamically based on user interaction, providing more control over when it is rendered.

For instance, there is a banking application, user clicks on 'Download Bank Statement' button.