PASSING DATA TO VIEWS

- Pass data to views using several approaches:

Strongly-typed data: viewmodel

Weakly-typed data

- ViewData (ViewDataAttribute)
- ViewBag

Strongly-typed data (viewmodel)

- The most robust approach is to specify a model type in the view.
- This model is commonly referred to as a viewmodel. Pass an instance of the viewmodel type to the view from the action.
- Specify a model using the @model directive. Use the model with @Model:

CONTD…

```
@model TrainingLab.ViewModels.Address
<h2>Contact</h2>
@Model.Name<br>
@Model.City, @Model.State <br>
```

To provide the model to the view, the controller passes it as a parameter:

CONTD…

```
public IActionResult Contact()
{
        ViewData["Message"] = "Your contact page.";
         var viewModel = new Address()
        {
                Name = “Binod",
                City = “Ktm",
                State = “3“
         };
     return View(viewModel);
}
```

CONTD…

```
public class Address
{
    public string Name
    { get; set; }
    public string City
    { get; set; }
    public string State
    { get; set; }
}
```

weakly-typed data (viewdata and viewbag)

- This collection can be referenced through either the **ViewData** or **ViewBag** properties on controllers and views.
- The **ViewBag** property is a wrapper around **ViewData** that provides dynamic properties for the underlying **ViewData** collection.
- **ViewData** and **ViewBag** are dynamically resolved at runtime.
- **ViewData** is a ViewDataDictionary object accessed through string keys.
- **String** data can be stored and used directly without the need for a cast, but you must cast other **ViewData** object values to specific types when you extract them.

CONTD…

```
public IActionResult SomeAction()
{
    ViewData["Greeting"] = "Hello";
    ViewData["Address"] = new Address()
    {   Name = “Binod",
        City = “Ktm",
        State = “3”
        };
    return View();
}
```

CONTD…

```
@{
    // Since Address isn't a string, it requires a cast.
    var address = ViewData["Address"] as Address;
}
@ViewData["Greeting"] World!
<address>
    @address.Name<br>
            @address.City, @address.State
</address>
```

ViewBag

- ViewBag isn't available in Razor Pages.
- **ViewBag** is a DynamicViewData object that provides dynamic access to the objects stored in **ViewData**.
- **ViewBag** can be more convenient to work with, since it doesn't require casting.

Partial View

- A **partial view** is a view that's rendered within another view.
- The HTML output generated by executing the partial view is rendered into the calling (or parent) view.
- Like views, partial views use the .cshtml file extension.
- Partial views are an effective way of breaking up large views into smaller components.
- They can reduce duplication of view content and allow view elements to be reused.

CONTD…

- @Html.PartialAsync("AuthorPartial")

Locate the view using relative paths

- @Html.PartialAsync("../Account/LoginPartial.cshtml")