ASP.NET Core - MVC Applications

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

Sharing Data from Controller to View

1. Passing Data to Views

In ASP.NET Core MVC, controllers often need to send data to views. There are multiple approaches for this:

1.1 ViewData

- **Definition:** A dictionary of key-value pairs (`ViewDataDictionary`) for passing data from a controller to a view.
- **Type:** `ViewData` is of type `ViewDataDictionary` and stores data as `object`.

- **Usage:**

```
// Controller
ViewData["Message"] = "Hello from ViewData";

// View
<p>@ViewData["Message"]</p>
```

- **Pros:** Simple to use, works with loosely typed data.
- **Cons:** Requires type casting, error-prone if keys are misspelled.

1.2 ViewBag

- **Definition:** A dynamic wrapper around `ViewData` that allows property-like access.
- **Type:** `dynamic` (runtime binding).
- **Usage:**

```
// Controller
ViewBag.Message = "Hello from ViewBag";

// View
<p>@ViewBag.Message</p>
```

- **Pros:** Cleaner syntax than `ViewData`.
- **Cons:** Also loosely typed, no compile-time checking.

1.3 TempData

- **Definition:** Used to store data **between two requests**. Data is stored in session or cookies temporarily.
- **Type:** `TempDataDictionary`
- **Usage:**

```
// Controller (Action 1)
TempData["Message"] = "Data for next request";
return RedirectToAction("NextAction");

// Controller (Action 2) or View
var message = TempData["Message"];
```

- **Pros:** Useful for redirect scenarios (e.g., after form submission).
- **Cons:** Data persists only for one request; must be type-cast.

1.4 Strongly Typed View

- **Definition:** Passing a model object directly to the view for strongly-typed access.
- **Usage:**

```
// Model
public class Student {
    public int Id { get; set; }
    public string Name { get; set; }
}

// Controller
public IActionResult Index() {
    var student = new Student { Id = 1, Name = "John" };
    return View(student);
}

// View (Index.cshtml)
@model Student
<p>@Model.Name</p>
```

✅ Summary Table

Feature	Scope	Lifetime	Type Safety	Typical Use
ViewData	Controller → View	Single request	❌	Small data, no model
ViewBag	Controller → View	Single request	❌	Quick passing of values
TempData	Across requests	One request	❌	Redirect scenarios
Strongly Typed View	Controller → View	Single request	✅	Full model data

Tag Helpers in View Development

2. Tag Helpers

- **Definition:** Tag Helpers are server-side components that generate HTML dynamically.
- **Syntax:** `<tag helper-attribute="value">`
- **Benefits:**
    - Cleaner syntax than HTML helpers
    - Strongly typed
    - IntelliSense support in Razor

Common Tag Helpers:

1. **Anchor Tag Helper**

```
<a asp-controller="Home" asp-action="Index">Home</a>
```

3. Input Tag Helper

```
<input asp-for="Email" class="form-control" />
```

4. Validation Tag Helpers

```
<span asp-validation-for="Name" class="text-danger"></span>
```

3. Handling Forms in Views

- Forms are used to collect user input and send it to controllers.
- **Steps:**
    1. Create a **model** to represent form data.
    2. Create a **view** with a form using `form` tag helper.
    3. Define **controller actions** to handle form submissions.

Example Model

```
public class Student {
    public string Name { get; set; }
    public int Age { get; set; }
}
```

Example View

```
<form asp-action="Create" method="post">
    <label>Name:</label>
    <input asp-for="Name" />

    <label>Age:</label>
    <input asp-for="Age" />

    <button type="submit">Submit</button>
</form>
```

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com