**State Management in ASP.NET Core MVC**

### **A Cookie Example**

Let’s create a new project and add a controller with endpoints to read and write values into cookies.

C#

|  |
| --- |
| **public class HomeController : Controller**  **{** |
| **public IActionResult Index()**  **{**  **//read cookie from Request object**  **string userName = Request.Cookies["UserName"];**  **return View("Index", userName);**  **}** |
| **[HttpPost]**  **public IActionResult Index(IFormCollection form)**  **{**  **string userName = form["userName"].ToString();**    **//set the key value in Cookie**  **CookieOptions option = new CookieOptions();**  **option.Expires = DateTime.Now.AddMinutes(10);**  **Response.Cookies.Append("UserName", userName, option);**    **return RedirectToAction(nameof(Index));**  **}** |
| **public IActionResult RemoveCookie()**  **{**  **//Delete the cookie**  **Response.Cookies.Delete("UserName");**  **return View("Index");**  **}**  **}** |

The Get version of the Index() method reads the UserName from the cookie and pass it to the view.

We use the Post version of the Index() method to get the value for userName from the form collection and assign it to the cookie.

For removing the cookie value, we use the RemoveCookie() endpoint.

Now let’s create the view:

C#

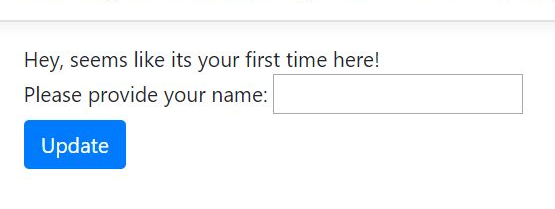
|  |
| --- |
| @**model string**    **@{**  **ViewData["Title"] = "Home Page";**  **}**    **@if (!string.IsNullOrWhiteSpace(Model))**  **{**  **@:<div>Welcome back, @Model</div>**  **@Html.ActionLink("Forget Me", "RemoveCookie")**  **}**  **else**  **{**  **@:**  **<form asp-action="Index">**  **<span>Hey, seems like it's your first time here!</span><br />**  **<label>Please provide your name:</label>**  **@Html.TextBox("userName")**  **<div class="form-group">**  **<input type="submit" value="Update" class="btn btn-primary" />**  **</div>**  **</**form>  } |

Here, we pass the UserName value into the View as the model.

If the UserName has a value, we greet the user by that name and give the user an option to forget the value by removing it from the cookie.

In case the UserName is empty, we show an input field for the user to enter his name and a submit button to update this in the cookie.

Now let’s run the application. Initially, the application asks the user to provide a name:



Once we provide a name and click update, the application greets us:



# Session State

We need to configure the session state before using it in our application.

**This can be done in the ConfigureServices() method in the Startup.cs class:**

|  |
| --- |
| services.AddSession(); |

Then, we need to enable session state in the Configure() method

|  |
| --- |
| app.UseSession(); |

***The order of configuration is important and we should invoke the UseSession() before invoking UseMVC().***

Let’s create a controller with endpoints to set and read a value from

|  |
| --- |
| public class WelcomeController : Controller  {      public IActionResult Index()      {          HttpContext.Session.SetString("Name", "John");          HttpContext.Session.SetInt32("Age", 32);            return View();      }        public IActionResult Get()      {          User newUser = new User()          {              Name = HttpContext.Session.GetString("Name"),              Age = HttpContext.Session.GetInt32("Age").Value          };            return View(newUser);      }  } |

**Now let’s navigate to /welcome/get:**

