<div class="event-card">

    <h3>@EventName</h3>

    <p>Date: @EventDate.ToString("MMMM dd, yyyy")</p>

    <p>Location: @EventLocation</p>

</div>

@code {

    [Parameter]

    public string EventName { get; set; }

    [Parameter]

    public DateTime EventDate { get; set; }

    [Parameter]

    public string EventLocation { get; set; }

}

public class Event

{

    public string Name { get; set; }

    public DateTime Date { get; set; }

    public string Location { get; set; }

}

@page "/events"

<h1>Upcoming Events</h1>

@foreach (var evt in Events)

{

    <EventCard EventName="@evt.Name" EventDate="@evt.Date" EventLocation="@evt.Location" />

}

@code {

    private List<Event> Events = new List<Event>

    {

        new Event { Name = "Corporate Meetup", Date = DateTime.Now.AddDays(10), Location = "New York" },

        new Event { Name = "Product Launch", Date = DateTime.Now.AddDays(20), Location = "San Francisco" },

        new Event { Name = "Annual Gala", Date = DateTime.Now.AddDays(30), Location = "Los Angeles" }

    };

}

<Router AppAssembly="@typeof(App).Assembly">

    <Found RouteData="@routeData">

        <PageTitle>EventEase</PageTitle>

        <LayoutView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />

    </Found>

    <NotFound>

        <h1>Page not found</h1>

    </NotFound>

</Router>

<nav>

    <a href="/events">Events</a>

    <a href="/registration">Register</a>

</nav>

<div class="event-card">

    <h3>@(string.IsNullOrEmpty(EventName) ? "Event Name Unavailable" : EventName)</h3>

    <p>Date: @(EventDate == DateTime.MinValue ? "Invalid Date" : EventDate.ToString("MMMM dd, yyyy"))</p>

    <p>Location: @(string.IsNullOrEmpty(EventLocation) ? "Location Unavailable" : EventLocation)</p>

</div>

@code {

    [Parameter]

    public string EventName { get; set; }

    [Parameter]

    public DateTime EventDate { get; set; }

    [Parameter]

    public string EventLocation { get; set; }

}

<Router AppAssembly="@typeof(App).Assembly">

    <Found Context="routeData">

        <LayoutView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />

    </Found>

    <NotFound>

        <h1>Page not found</h1>

        <p>The page you're looking for doesn't exist. <a href="/events">Go back to events</a></p>

    </NotFound>

</Router>

<h1>Upcoming Events</h1>

<Virtualize Items="Events" Context="eventItem">

    <EventCard EventName="@eventItem.Name" EventDate="@eventItem.Date" EventLocation="@eventItem.Location" />

</Virtualize>

@code {

    private List<Event> Events = new List<Event>();

    protected override void OnInitialized()

    {

        // Simulate large dataset

        for (int i = 1; i <= 1000; i++)

        {

            Events.Add(new Event

            {

                Name = $"Event {i}",

                Date = DateTime.Now.AddDays(i),

                Location = $"Location {i}"

            });

        }

    }

}

<h2>Event Registration</h2>

<EditForm Model="@UserRegistration" OnValidSubmit="HandleValidSubmit">

<DataAnnotationsValidator />

<ValidationSummary />

<div>

<label>Name:</label>

<InputText @bind-Value="UserRegistration.Name" />

<ValidationMessage For="@(() => UserRegistration.Name)" />

</div>

<div>

<label>Email:</label>

<InputText @bind-Value="UserRegistration.Email" />

<ValidationMessage For="@(() => UserRegistration.Email)" />

</div>

<button type="submit">Register</button>

</EditForm>

<p>@RegistrationStatus</p>

@code {

private UserRegistrationModel UserRegistration = new();

private string RegistrationStatus;

private void HandleValidSubmit()

{

RegistrationStatus = "Registration successful!";

// Save user data or perform additional actions

}

public class UserRegistrationModel

{

[Required]

public string Name { get; set; }

[Required]

[EmailAddress]

public string Email { get; set; }

}

}

public class UserSessionService

{

public string UserName { get; set; }

public string UserEmail { get; set; }

public bool IsRegistered { get; set; } = false;

public void RegisterUser(string name, string email)

{

UserName = name;

UserEmail = email;

IsRegistered = true;

}

}

builder.Services.AddSingleton<UserSessionService>();

@inject UserSessionService SessionService

@code {

private void HandleValidSubmit()

{

SessionService.RegisterUser(UserRegistration.Name, UserRegistration.Email);

RegistrationStatus = "Registration successful!";

}

}

<h2>My Events</h2>

@if (SessionService.IsRegistered)

{

<ul>

@foreach (var eventItem in RegisteredEvents)

{

<li>@eventItem</li>

}

</ul>

<button @onclick="AddEvent">Add Event</button>

}

else

{

<p>Please register to track attendance.</p>

}

@code {

[Inject]

private UserSessionService SessionService { get; set; }

private List<string> RegisteredEvents = new();

private void AddEvent()

{

RegisteredEvents.Add($"Event {RegisteredEvents.Count + 1}");

}

}