# **RazorPages PracticeEx**

### **B1:** Kết nối database SQL

### **B2:** Tạo blank solution, class library repo, class library services

### **B3:** Add package cho repo (bấm vào class repo)

<ItemGroup>   
 <PackageReference Include="Microsoft.EntityFrameworkCore" Version="8.0.2" />   
 <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.2" />   
 <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.2">   
 <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>   
 <PrivateAssets>all</PrivateAssets>   
 </PackageReference>   
 <PackageReference Include="Microsoft.Extensions.Configuration" Version="8.0.0" />   
 <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="8.0.0" />   
</ItemGroup>

### **B4:** EF Tool database sql

### **B5:** Sau khi đã có model, chuyển dbcontext qua thư mục DBContext (đổi namespace)

### **B6:** Trong DBContext Note connection string có sẵn và ngay trên đó thay bằng:

public static string GetConnectionString(string connectionStringName)   
{   
 var config = new ConfigurationBuilder()   
 .SetBasePath(AppDomain.CurrentDomain.BaseDirectory)   
 .AddJsonFile("appsettings.json")   
 .Build();  
  
 string connectionString = config.GetConnectionString(connectionStringName);   
 return connectionString;   
}  
  
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)   
{   
 optionsBuilder.UseSqlServer(GetConnectionString("DefaultConnection")).UseQueryTrackingBehavior(QueryTrackingBehavior.NoTracking);   
}

### **B7:** Tạo folder Basic trong repo bên trong tạo class GenericRepository:

public class GenericRepository<T> where T : class   
{   
 protected SU25LionDBContext \_context;  
  
 public GenericRepository()   
 {   
 \_context ??= new SU25LionDBContext();   
 }  
  
 public GenericRepository(SU25LionDBContext context)   
 {   
 \_context = context;   
 }  
  
 public List<T> GetAll()   
 {   
 return \_context.Set<T>().ToList();   
 }  
  
 public async Task<List<T>> GetAllAsync()   
 {   
 return await \_context.Set<T>().ToListAsync();   
 }  
  
 public void Create(T entity)   
 {   
 \_context.Add(entity);   
 \_context.SaveChanges();   
 }  
  
 public async Task<int> CreateAsync(T entity)   
 {   
 \_context.Add(entity);   
 return await \_context.SaveChangesAsync();   
 }  
  
 public void Update(T entity)   
 {   
 //// Turning off Tracking for UpdateAsync in Entity Framework   
 \_context.ChangeTracker.Clear();   
 var tracker = \_context.Attach(entity);   
 tracker.State = EntityState.Modified;   
 \_context.SaveChanges();   
 }  
  
 public async Task<int> UpdateAsync(T entity)   
 {   
 //// Turning off Tracking for UpdateAsync in Entity Framework   
 \_context.ChangeTracker.Clear();   
 var tracker = \_context.Attach(entity);   
 tracker.State = EntityState.Modified;   
 return await \_context.SaveChangesAsync();   
 }  
  
 public bool Remove(T entity)   
 {   
 \_context.Remove(entity);   
 \_context.SaveChanges();   
 return true;   
 }  
  
 public async Task<bool> RemoveAsync(T entity)   
 {   
 \_context.Remove(entity);   
 await \_context.SaveChangesAsync();   
 return true;   
 }  
  
 public T GetById(int id)   
 {   
 return \_context.Set<T>().Find(id);   
 }  
  
 public async Task<T> GetByIdAsync(int id)   
 {   
 return await \_context.Set<T>().FindAsync(id);   
 }  
  
 public T GetById(string code)   
 {   
 return \_context.Set<T>().Find(code);   
 }  
  
 public async Task<T> GetByIdAsync(string code)   
 {   
 return await \_context.Set<T>().FindAsync(code);   
 }  
  
 public T GetById(Guid code)   
 {   
 return \_context.Set<T>().Find(code);   
 }  
  
 public async Task<T> GetByIdAsync(Guid code)   
 {   
 return await \_context.Set<T>().FindAsync(code);   
 }   
}

### **B8:** Tạo repo cho các thực thể

### **B9:** Tạo Service và IService cho các repo

### **B10:** Set reference project

### **B11**: Set up program cs

**Ngay dưới AddRazorPage**

builder.Services.AddScoped<ILionProfileService, LionProfileService>();   
builder.Services.AddScoped<ILionTypeService, LionTypeService>();   
builder.Services.AddScoped<ILionAccountService, LionAccountService>();   
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)   
 .AddCookie(options =>   
 {   
 options.LoginPath = "/Login";   
 options.AccessDeniedPath = "/Forbidden";   
 options.ExpireTimeSpan = TimeSpan.FromMinutes(5);   
 });

**Dưới MapRazorPage**

app.MapRazorPages().RequireAuthorization();

### **B12**: Set startup project

### **B13**: Trong folder Page tạo folder tên thực thể chính có s

### **B14:** Tạo các page CRUD theo Repo thực thể chính Models

### **B15:** Copy folder Account của thầy qua Page và sửa tất cả các namespace ở view và code

### **B16:** Đóng tất cả các code ở CUD (chừa BindProperties ra) và chừa lại index

### **B17:** Sửa appsetting thêm dấu , và sau dấu , thêm:

"ConnectionStrings": {   
 "DefaultConnection": "Data Source=LAPTOP-GFKPMHK\\\\SQLEXPRESS;Initial Catalog=SU25LionDB;Persist Security Info=True;User ID=SA;Password=12345;TrustServerCertificate=True"   
}

### **B18:** Sửa Index code:

\[Authorize(Roles \= "2,3")\]  
  
public class IndexModel : PageModel  
  
{  
  
 private readonly ILionProfileService \\_lionProfileService;  
  
 public IndexModel(ILionProfileService lionProfileService)  
  
 {  
  
 \\_lionProfileService \= lionProfileService;  
  
 }  
  
 public IList\<LionProfile\> LionProfile { get; set; } \= default\!;  
  
 \[BindProperty(SupportsGet \= true)\]  
  
 public int PageNumber { get; set; } \= 1;  
  
 public int PageSize { get; set; } \= 3;  
  
 public int TotalPages { get; set; }  
  
 public async Task OnGetAsync()  
  
 {  
  
 var allItems \= await \\_lionProfileService.GetAllAsync();  
  
 int totalItems \= allItems.Count;  
  
 TotalPages \= (int)Math.Ceiling(totalItems / (double)PageSize);  
  
 LionProfile \= allItems  
  
 .Skip((PageNumber \- 1\) \\* PageSize)  
  
 .Take(PageSize)  
  
 .ToList();  
  
 }  
  
}

### **Trong cshtml index phân trang:**

\<nav\>  
  
 \<ul class="pagination justify-content-center"\>  
  
 \<li class="page-item @(Model.PageNumber \== 1 ? "disabled" : "")"\>  
  
 \<a class="page-link" asp-route-PageNumber="@(Model.PageNumber \- 1)"\>Previous\</a\>  
  
 \</li\>  
  
 @for (int i \= 1; i \<= Model.TotalPages; i++)  
  
 {  
  
 \<li class="page-item @(i \== Model.PageNumber ? "active" : "")"\>  
  
 \<a class="page-link" asp-route-PageNumber="@i"\>@i\</a\>  
  
 \</li\>  
  
 }  
  
 \<li class="page-item @(Model.PageNumber \== Model.TotalPages ? "disabled" : "")"\>  
  
 \<a class="page-link" asp-route-PageNumber="@(Model.PageNumber \+ 1)"\>Next\</a\>  
  
 \</li\>  
  
 \</ul\>  
  
\</nav\>

### **B19**: Sửa details

\[Authorize(Roles \= "2,3")\]  
  
public class DetailsModel : PageModel  
  
{  
  
 private readonly ILionProfileService \\_lionProfileService;  
  
 public DetailsModel(ILionProfileService lionProfileService)  
  
 {  
  
 \\_lionProfileService \= lionProfileService;  
  
 }  
  
 public LionProfile LionProfile { get; set; } \= default\!;  
  
 public async Task\<IActionResult\> OnGetAsync(int? id)  
  
 {  
  
 if (id \== null)  
  
 {  
  
 return NotFound();  
  
 }  
  
 var lionprofile \= await \\_lionProfileService.GetByIdAsync(id.Value);  
  
 if (lionprofile \== null)  
  
 {  
  
 return NotFound();  
  
 }  
  
 else  
  
 {  
  
 LionProfile \= lionprofile;  
  
 }  
  
 return Page();  
  
 }  
  
}

### **B20**: Sửa Create

[Authorize(Roles = "2")]   
public class CreateModel : PageModel   
{   
 private readonly ILionProfileService \_lionProfileService;   
 private readonly ILionTypeService \_lionTypeService;  
  
 public CreateModel(ILionProfileService lionProfileService, ILionTypeService lionTypeService)   
 {   
 \_lionProfileService = lionProfileService;   
 \_lionTypeService = lionTypeService;   
 }  
  
 public async Task<IActionResult> OnGetAsync()   
 {   
 var LionType = await \_lionTypeService.GetAllAsync();   
 ViewData["LionTypeId"] = new SelectList(LionType, "LionTypeId", "LionTypeName");   
 return Page();   
 }  
  
 [BindProperty]   
 public LionProfile LionProfile { get; set; } = default!;   
   
 public async Task<IActionResult> OnPostAsync()   
 {   
 if (!ModelState.IsValid)   
 {   
 return Page();   
 }  
  
 var result = await \_lionProfileService.CreateAsync(LionProfile);   
 if (result == null)   
 {   
 ModelState.AddModelError(string.Empty, "Unable to create the lion profile, Please try again.");   
 return Page();   
 }  
  
 return RedirectToPage("./Index");   
 }   
}

#### **Validate entity chính trong model**

[Required(ErrorMessage = "Lion Type is required.")]   
[Required]   
[MinLength(4)]   
[RegularExpression(@"^([A-Z][a-z0-9]\*)(\s[A-Z][a-z0-9]\*)\*$", ErrorMessage = "Each word must start with a capital letter and contain no special characters.")]   
[Range(30.01, double.MaxValue, ErrorMessage = "Weight must be greater than 30.")]

### **B21:** Sửa Edit

[Authorize(Roles = "2")]   
public class EditModel : PageModel   
{   
 private readonly ILionProfileService \_lionProfileService;   
 private readonly ILionTypeService \_lionTypeService;  
  
 public EditModel(ILionProfileService lionProfileService, ILionTypeService lionTypeService)   
 {   
 \_lionProfileService = lionProfileService;   
 \_lionTypeService = lionTypeService;   
 }  
  
 [BindProperty]   
 public LionProfile LionProfile { get; set; } = default!;  
  
 public async Task<IActionResult> OnGetAsync(int? id)   
 {   
 if (id == null)   
 {   
 return NotFound();   
 }  
  
 var lionProfile = await \_lionProfileService.GetByIdAsync(id.Value);   
 if (lionProfile == null)   
 {   
 return NotFound();   
 }  
  
 LionProfile = lionProfile;   
 var LionType = await \_lionTypeService.GetAllAsync();   
 ViewData["LionTypeId"] = new SelectList(LionType, "LionTypeId", "LionTypeName");   
 return Page();   
 }  
  
 // To protect from overposting attacks, enable the specific properties you   
 // want to bind to.   
 // For more information, see https://aka.ms/RazorPagesCRUD.   
 public async Task<IActionResult> OnPostAsync()   
 {   
 if (!ModelState.IsValid)   
 {   
 return Page();   
 }  
  
 try   
 {   
 await \_lionProfileService.UpdateAsync(LionProfile);   
 }   
 catch (DbUpdateConcurrencyException)   
 {   
 if (!await LionProfileExists(LionProfile.LionProfileId))   
 {   
 return NotFound();   
 }   
 else   
 {   
 throw;   
 }   
 }  
  
 return RedirectToPage("./Index");   
 }  
  
 private async Task<bool> LionProfileExists(int id)   
 {   
 var lionProfile = await \_lionProfileService.GetByIdAsync(id);   
 return lionProfile != null && lionProfile.LionProfileId == id;   
 }   
}

### **B22:** Sửa delete

[Authorize(Roles = "2")]   
public class DeleteModel : PageModel   
{   
 private readonly ILionProfileService \_lionProfileService;   
 private readonly ILionTypeService \_lionTypeService;   
 public DeleteModel(ILionProfileService lionProfileService, ILionTypeService lionTypeService)   
 {   
 \_lionProfileService = lionProfileService;   
 \_lionTypeService = lionTypeService;   
 }  
  
 [BindProperty]   
 public LionProfile LionProfile { get; set; } = default!;  
  
 public async Task<IActionResult> OnGetAsync(int? id)   
 {   
 if (id == null)   
 {   
 return NotFound();   
 }  
  
 var lionprofile = await \_lionProfileService.GetByIdAsync(id.Value);   
 if (lionprofile == null)   
 {   
 return NotFound();   
 }   
 else   
 {   
 LionProfile = lionprofile;   
 }   
 return Page();   
 }  
  
 public async Task<IActionResult> OnPostAsync(int? id)   
 {   
 if (id == null)   
 {   
 return NotFound();   
 }  
  
 var lionprofile = await \_lionProfileService.GetByIdAsync(id.Value);   
 if (lionprofile != null)   
 {   
 await \_lionProfileService.DeleteAsync(lionprofile.LionProfileId);   
 }   
 return RedirectToPage("./Index");   
 }   
}

### **B23**: Copy index ra đổi tên thành search (đổi thành SearchModel)

### **B24:** Code Search

[Authorize(Roles = "2,3")]   
public class SearchModel : PageModel   
{   
 private readonly ILionProfileService \_lionProfileService;  
  
 public SearchModel(ILionProfileService lionProfileService)   
 {   
 \_lionProfileService = lionProfileService;   
 }  
  
 public IList<LionProfile> LionProfile { get; set; } = default!;  
  
 [BindProperty(SupportsGet = true)]   
 public int PageNumber { get; set; } = 1;  
  
 public int PageSize { get; set; } = 3;   
 public int TotalPages { get; set; }  
  
 [BindProperty(SupportsGet = true)]   
 public string? LionTypeName { get; set; }  
  
 [BindProperty(SupportsGet = true)]   
 public double? Weight { get; set; }  
  
 public async Task OnGetAsync()   
 {   
 var allItems = await \_lionProfileService.SearchAsync(LionTypeName, Weight ?? 0);   
 int totalItems = allItems.Count;   
 TotalPages = (int)Math.Ceiling(totalItems / (double)PageSize);  
  
 LionProfile = allItems   
 .Skip((PageNumber - 1) \* PageSize)   
 .Take(PageSize)   
 .ToList();   
 }   
}

### **B25:** Cshtml Search

@page "/LionProfile/Search"   
@model LionPetManagement\_HoangQuocViet.Pages.LionProfiles.SearchModel  
  
@{   
}  
  
<form method="get" class="row mb-3">   
 <div class="col-md-3">   
 <input type="text" name="LionTypeName" class="form-control" placeholder="LionTypeName" value="@Model.LionTypeName" />   
 </div>   
 <div class="col-md-3">   
 <input type="number" name="Weight" class="form-control" placeholder="Weight" value="@Model.Weight" />   
 </div>   
 <button class="col-md-3" type="submit" class="btn btn-primary w-100">Search</button>   
</form>  
  
<table class="table">   
 <thead>   
 <tr>   
 <th>   
 @Html.DisplayNameFor(model => model.LionProfile[0].LionName)   
 </th>   
 <th>   
 @Html.DisplayNameFor(model => model.LionProfile[0].Weight)   
 </th>   
 <th>   
 @Html.DisplayNameFor(model => model.LionProfile[0].Characteristics)   
 </th>   
 <th>   
 @Html.DisplayNameFor(model => model.LionProfile[0].Warning)   
 </th>   
 <th>   
 @Html.DisplayNameFor(model => model.LionProfile[0].ModifiedDate)   
 </th>   
 <th>   
 @Html.DisplayNameFor(model => model.LionProfile[0].LionType.LionTypeName)   
 </th>   
 <th></th>   
 </tr>   
 </thead>   
 <tbody id="LionProfileId">   
 @if (Model.LionProfile != null && Model.LionProfile.Count > 0)   
 {   
 foreach (var item in Model.LionProfile)   
 {   
 <tr id="@item.LionProfileId">   
 <td>   
 @Html.DisplayFor(modelItem => item.LionName)   
 </td>   
 <td>   
 @Html.DisplayFor(modelItem => item.Weight)   
 </td>   
 <td>   
 @Html.DisplayFor(modelItem => item.Characteristics)   
 </td>   
 <td>   
 @Html.DisplayFor(modelItem => item.Warning)   
 </td>   
 <td>   
 @Html.DisplayFor(modelItem => item.ModifiedDate)   
 </td>   
 <td>   
 @Html.DisplayFor(modelItem => item.LionType.LionTypeName)   
 </td>   
 <td>   
 <a asp-page="./Edit" asp-route-id="@item.LionProfileId">Update</a> |   
 <a asp-page="./Details" asp-route-id="@item.LionProfileId">Detail</a> |   
 <a asp-page="./Delete" asp-route-id="@item.LionProfileId">Delete</a>   
 </td>   
 </tr>   
 }   
 }   
 else   
 {   
 <tr>   
 <td colspan="8" class="text-center text-muted">   
 <i class="fas fa-info-circle me-1"></i> Không có dữ liệu để hiển thị.   
 </td>   
 </tr>   
 }   
 </tbody>   
</table>   
<nav>   
 <ul class="pagination justify-content-center">   
 <li class="page-item @(Model.PageNumber == 1 ? "disabled" : "")">   
 <a class="page-link" asp-route-PageNumber="@(Model.PageNumber - 1)">Previous</a>   
 </li>   
 @for (int i = 1; i <= Model.TotalPages; i++)   
 {   
 <li class="page-item @(i == Model.PageNumber ? "active" : "")">   
 <a class="page-link" asp-route-PageNumber="@i">@i</a>   
 </li>   
 }   
 <li class="page-item @(Model.PageNumber == Model.TotalPages ? "disabled" : "")">   
 <a class="page-link" asp-route-PageNumber="@(Model.PageNumber + 1)">Next</a>   
 </li>   
 </ul>   
</nav>

### **B26** Thêm folder hub trong razor page (nếu kịp)

### **B27** Add cái Hub vào [Program.cs](http://Program.cs)

### builder.Services.AddSignalR();

#### **và dưới RequireAuthentication**

app.MapHub<LionPetManagementHub>("/LionPetManagementHub");

### **B27** Tạo class xxxHub

public class LionPetManagementHub : Hub   
{   
 private readonly ILionProfileService \_lionProfileService;   
 private readonly ILionTypeService \_lionTypeService;  
  
 public LionPetManagementHub(ILionProfileService lionProfileService, ILionTypeService lionTypeService)   
 {   
 \_lionProfileService = lionProfileService;   
 \_lionTypeService = lionTypeService;   
 }  
  
 public async Task HubDelete\_LionPetManagementHub(string lionProfileId)   
 {   
 await Clients.All.SendAsync("Receiver\_DeleteLionPetManagementHub", lionProfileId);   
 await \_lionProfileService.DeleteAsync(int.Parse(lionProfileId));   
 }   
}  
  
### \*\*B28\*\*: Nếu là delete signal R thì trong delete cshtml ngay end thêm  
  
<script>   
 function confirmDeleteForm() {   
 return confirm("Confirm?");   
 }   
</script>   
<script src="~/js/signalr/dist/browser/signalr.js"></script>   
<script>   
 "use strict";  
  
 var connection = new signalR.HubConnectionBuilder().withUrl("/LionPetManagementHub").build();  
  
 connection.start().then(function () {   
 console.log("Connected to the SignalR Hub");   
 }).catch(function (err) {   
 return console.error(err.toString());   
 });  
  
 document.getElementById("btnHubDelete").addEventListener("click", function (event) {   
 event.preventDefault(); // Ngăn form gửi đi ngay  
  
 // Xác nhận trước khi xóa   
 var confirmed = confirm("Confirm?");   
 if (!confirmed) return;  
  
 var dataObj = document.getElementById("LionProfile\_LionProfileId").value;   
 console.log(dataObj);  
  
 connection.invoke("HubDelete\_LionPetManagementHub", dataObj).catch(function (err) {   
 return console.error(err.toString());   
 });   
 event.preventDefault();   
 });   
</script>  
  
Và nút   
<input type="button" id="btnHubDelete" value="Delete by SignalRHub" class="btn btn-primary" />

### **B29**: Ở Index.cshtml dưới cùng thêm

<script src="~/js/signalr/dist/browser/signalr.js"></script>   
<script>   
 "use strict";  
  
 var connection = new signalR.HubConnectionBuilder().withUrl("/LionPetManagementHub").build();  
  
 connection.start().then(function () {   
 console.log("Connected to the SignalR Hub");   
 }).catch(function (err) {   
 return console.error(err.toString());   
 });  
  
 connection.on("Receiver\_DeleteLionPetManagementHub", function (LionProfileId) {   
 $("#LionProfileId").find(tr[id='${LionProfileId}']).remove();   
 });   
</script>