.NET 9 App Dev Hands-On Workshop

Blazor Lab 7 – Car Pages

This lab finishes the client UI by adding the Car pages. Before starting this lab, you must have completed Blazor Lab 6.

Part 1: Add the Car Pages

Step 1: The Base Component

- Add a new directory named `Cars` to the `Pages` directory. Under that directory, add a new directory named `Base`. To that directory, add a new Razor component named `CarBase.razor`. Update it to the following:

```
@code {
  [Inject]
  protected ICarDataService CarDataServiceInstance { get; set; }
  [Inject]
  protected NavigationManager NavManagerInstance { get; set; }
  protected Car Entity = default;
}
```

- Add the following to the `_Imports.razor` file:

```
@using AutoLot.Blazor.Pages.Cars
@using AutoLot.Blazor.Pages.Cars.Base
```

Step 2: The Create Page

- Add a new Razor component named `Create.razor` to the `Cars` folder. Clear out the contents and update it to the following:

```
@page "/cars/create"
@inherits CarBase
<PageTitle>Create Vehicle</PageTitle>
<h1>Create a New Car</h1>
@if (Entity == null || !_makes.Any())
{
  <div><em>Loading...</em></div>
}
else
{
  <EditForm Model="Entity" OnSubmit="AddCarAsync">
    <CarEdit CarInstance="Entity" Makes="_makes" />
    <div class="pt-4">
      <button class="btn btn-success">Create <i class="fa-solid fa-plus"></i></button>
      | <ListHelper RouteStart="cars" />
    </div>
  </EditForm>
}
```

```
@code {
  [Inject] private IMakeDataService _makeDataService { get; set; }
  public async Task AddCarAsync(EditContext context)
  {
    if (context.Validate())
    {
      var car = await CarDataServiceInstance.AddEntityAsync(Entity);
      NavManagerInstance.NavigateTo($"/cars/details/{car.Id}");
    }
  }
  public override async Task SetParametersAsync(ParameterView parameters)
  {
    await base.SetParametersAsync(parameters);
    Entity = new Car();
    _makes = await _makeDataService.GetAllEntitiesAsync();
    StateHasChanged();
  }
  private List<Make> _makes = [];
}
```

Step 3: The Delete Page

- Add a new Razor component named `Delete.razor` to the `Cars` folder. Clear out the contents and update it to the following:

```
@page "/cars/delete/{Id:int}"
@inherits CarBase
<h1>Delete Vehicle</h1>
@if (Entity == null)
{
  <div><em>Loading...</em></div>
  <PageTitle>Delete Vehicle</PageTitle>
}
else
{
  <PageTitle>Delete @Entity.PetName</PageTitle>
  <CarDetail CarInstance="Entity" />
  <EditForm Model="Entity" OnSubmit="DeleteCarAsync">
    <div class="pt-4">
      <button class="btn btn-danger">Delete <i class="fa-solid fa-trash"></i></button>
      | <ListHelper RouteStart="cars" />
    </div>
  </EditForm>
}
```

All files copyright Phil Japikse (http://www.skimedic.com/blog)

```
@code {
  [Parameter] public int Id { get; set; }
  public async Task DeleteCarAsync(EditContext context)
  {
    if (context.Validate())
    {
      await CarDataServiceInstance.DeleteEntityAsync((Car)context.Model);
    }
    NavManagerInstance.NavigateTo("/cars/index");
  }
  public override async Task SetParametersAsync(ParameterView parameters)
  {
    await base.SetParametersAsync(parameters);
    Entity = await CarDataServiceInstance.GetEntityAsync(Id);
    StateHasChanged();
  }
}
```

Step 4: The Details Page

- Add a new Razor component named `Details.razor` to the `Cars` folder. Clear out the contents and update it to the following:

```
@page "/cars/details/{Id:int}"
@inherits CarBase
<h1>Vehicle Details</h1>
@if (Entity == null)
{
  <div><em>Loading...</em></div>
  <PageTitle>Vehicle Details</PageTitle>
}
else
{
  <PageTitle>@Entity.PetName Details</PageTitle>
  <CarDetail CarInstance="Entity" />
  <EditHelper Id="@Entity.Id" RouteStart="cars" />
  <DeleteHelper Id="@Entity.Id" RouteStart="cars" />
  <ListHelper RouteStart="cars" />
}
@code {
  [Parameter]
  public int Id { get; set; }
  public override async Task SetParametersAsync(ParameterView parameters)
  {
    await base.SetParametersAsync(parameters);
    Entity = await CarDataServiceInstance.GetEntityAsync(Id);
    StateHasChanged();
  }
}
```

All files copyright Phil Japikse (http://www.skimedic.com/blog)

Step 5: The Edit Page

- Add a new Razor component named `Edit.razor` to the `Cars` folder. Clear out the contents and update it to the following:

```
@page "/cars/edit/{Id:int}"
@inherits CarBase
<h1>Edit Vehicle</h1>
@if (Entity == null || !_makes.Any())
{
  <div><em>Loading...</em></div>
  <PageTitle>Edit Vehicle</PageTitle>
}
else
{
  <PageTitle>Edit @Entity.PetName</PageTitle>
  <h1>Edit Details for @Entity.PetName</h1>
  <EditForm Model="Entity" OnSubmit="SaveCarAsync">
    <CarEdit CarInstance="Entity" Makes="_makes" />
    <div class="pt-4">
      <button class="btn btn-primary" disabled="@disabled">
           Save <i class="fa-solid fa-save"></i></button>
      | <ListHelper RouteStart="cars" />
    </div>
    <div class="pt-2">@savingMessage</div>
  </EditForm>
}
@code {
  [Parameter]
  public int Id { get; set; }
  [Inject]
  private IMakeDataService _makeDataService { get; set; }
  private bool disabled = false;
  private string savingMessage = "";
  public async Task SaveCarAsync(EditContext context)
  {
    if (context.Validate())
    {
      disabled = true;
      savingMessage = "Saving (with a built-in delay)...";
      await Task.Delay(5000);
      await CarDataServiceInstance.UpdateEntityAsync(Id,Entity);
      NavManagerInstance.NavigateTo($"/cars/details/{Id}");
    }
  }
  public override async Task SetParametersAsync(ParameterView parameters)
  {
    await base.SetParametersAsync(parameters);
    Entity = await CarDataServiceInstance.GetEntityAsync(Id);
    _makes = await _makeDataService.GetAllEntitiesAsync();
    StateHasChanged();
  }
  private List<Make> _makes = [];
}
```

All files copyright Phil Japikse (http://www.skimedic.com/blog)

Step 6: The Index Page

- Add a new Razor component named `Index.razor` to the `Cars` folder. Clear out the contents and update it to the following:

```
@page "/cars/index"
@page "/cars/index/{MakeId:int}/{MakeName}"
@inherits CarBase
<h1>Vehicle Inventory</h1>
@if (!MakeName.Equals("All", StringComparison.OrdinalIgnoreCase))
{
  <h3>@MakeName</h3>
  <PageTitle>@MakeName Inventory</PageTitle>
}
else
{
  <PageTitle>Vehicle Inventory</PageTitle>
}
<CreateHelper RouteStart="cars"></CreateHelper>
<p />
@if (!_cars.Any())
{
  <div><em>Loading...</em></div>
}
else
{
  <CarList Cars="_cars" />
}

@code {
  [Parameter]
  public int? MakeId { get; set; }
  [Parameter]
  public string MakeName { get; set; }
  public override async Task SetParametersAsync(ParameterView parameters)
  {
    await base.SetParametersAsync(parameters);
    MakeName ??= "All";
    _cars = MakeId is > 0
        ? await CarDataServiceInstance.GetByMakeAsync(MakeId.Value)
        : await CarDataServiceInstance.GetAllEntitiesAsync();
    StateHasChanged();
  }
  private List<Car> _cars = [];
}
```

Run the app and test all of the Car pages.

Summary

This lab added the Car pages and completed the client pages.

All files copyright Phil Japikse (http://www.skimedic.com/blog)

Next Steps

The following lab will use JS Interop to save the application state in the browser.

All files copyright Phil Japikse (http://www.skimedic.com/blog)