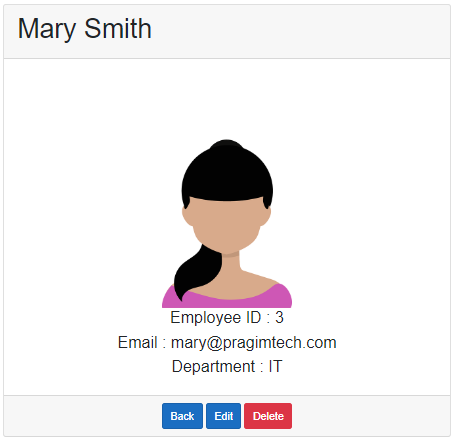
Display data from two or more tables in Blazor

In this video we will discuss how to retrieve data from two or more related database tables and display them in a Blazor web application. There are several ways to do this. There is no right or wrong way. Which way you chose really depends on the architecture, complexity and the requirements of your application.



As you can see from the image, we are displaying the following employee details.

* Id
* Name
* Email
* Department Name

Id, Name, and Email come from the Employees database table and the Department Name comes from the Departments table. Here we are working with two database tables, but you can use the same technique to work with three or even more database tables.

Employee Class

The Department property carries the employee Department data i.e DepartmentId and DepartmentName.

public class Employee

{

public int EmployeeId { get; set; }

[Required]

[MinLength(2)]

public string FirstName { get; set; }

[Required]

public string LastName { get; set; }

public string Email { get; set; }

public DateTime DateOfBrith { get; set; }

public Gender Gender { get; set; }

public int DepartmentId { get; set; }

public Department Department { get; set; }

public string PhotoPath { get; set; }

}

Copy

EmployeeRepository

In the Employee entity, to include related entity data like Department, chain the Include() method. Chain ThenInclude() method if you want to include additional related entities like skills data, experience data etc.

public class EmployeeRepository : IEmployeeRepository

{

private readonly AppDbContext appDbContext;

public EmployeeRepository(AppDbContext appDbContext)

{

this.appDbContext = appDbContext;

}

public async Task<Employee> GetEmployee(int employeeId)

{

return await appDbContext.Employees

.Include(e => e.Department)

.FirstOrDefaultAsync(e => e.EmployeeId == employeeId);

}

}

Copy

EmployeeDetails Component

@page "/employeedetails/{id}"

@page "/employeedetails"

@inherits EmployeeDetailsBase

@if (Employee == null || Employee.Department == null)

{

<div class="spinner"></div>

}

else

{

<div class="row justify-content-center m-3">

<div class="col-sm-8">

<div class="card">

<div class="card-header">

<h1>@Employee.FirstName @Employee.LastName</h1>

</div>

<div class="card-body text-center">

<img class="card-img-top" src="@Employee.PhotoPath" />

<h4>Employee ID : @Employee.EmployeeId</h4>

<h4>Email : @Employee.Email</h4>

<h4>Department : @Employee.Department.DepartmentName</h4>

</div>

<div class="card-footer text-center">

<a href="/" class="btn btn-primary">Back</a>

<a href="#" class="btn btn-primary">Edit</a>

<a href="#" class="btn btn-danger">Delete</a>

</div>

</div>

</div>

</div>

}

Copy

The following null check is required. Otherwise you will get a NullReferenceException, if either Employee or Department properties are not initialized by the time the view references those properties.

@if (Employee == null || Employee.Department == null)

{

<div class="spinner"></div>

}