Project: Display the employee under each country

90

Inverse Property

- Completing a one-to-many relationship requires a inverse property to be added to the class at the "one" end of the relationship.

```
public class Country
{
    public int CountryID { get; set; }
    public string CountryName { get; set; }
    public IEnumerable<Employee> Employees { get; set; }//反向属性
}
```

91

Project: Add inverse property to Country class

```
public class Employee
{
    public int EmployeeId { get; set; }
    public string EmployeeName { get; set; }
    public string Title { get; set; }
    public int CountryID { get; set; } //外键属性
    public Country Country { get; set; } //导航属性
}
```

```
public class Country
{
    public int CountryID { get; set; }
    public string CountryName { get; set; }
    public IEnumerable<Employee> Employees { get; set; } //反向属性
}
```

92

Project: Update the repository

```
public List<Country> SelectAll()
{
    return context.Countries.Include(c => c.Employees).ToList();
}
```

93

Project: Update the view

```
<table>
    <tr>
        <td>Country ID</td>
        <td>Country Name</td>
    </tr>
    @foreach (Country c in Model)
    {
        <tr>
            <td>@c.CountryID</td>
            <td>@c.CountryName</td>
        </tr>
        @if(c.Employees!=null)
        {
            @foreach(var e in c.Employees)
            {
                <tr style="background-color:lightgray">
                    <td></td>
                    <td>@e.EmployeeName</td>
                </tr>
            }       }    }
</table>
```

94

Project: Run the application

95

One-to-One Relationship

- Add reciprocal navigation properties and add a foreign key property to the dependent entity class.

```
public class Employee
{
    public int EmployeeId { get; set; }
    public string EmployeeName { get; set; }
    public string Title { get; set; }
    public int CountryID { get; set; }
    public Country Country { get; set; }
    public Contact Contact { get; set; }
}
```

```
public class Contact
{
    public int ContactId { get; set; }
    public string Phone { get; set; }
    public string Address { get; set; }
    public int EmployeeId { get; set; }
    public Employee Employee { get; set; }
}
```


Project: Update the repository

```
public List<Employee> SelectAll()
{
    return context.Employees.Include(e => e.Country).
           Include(e=>e.Contact).ToList();
}
```


Project: Update the view

```
@foreach (Employee e in Model)
{
    <tr>
        <td>@e.EmployeeId</td>
        <td>@e.EmployeeName</td>
        <td>@e.Title</td>
        <td>@e.Country.CountryName</td>
        <td>@e.Contact.Phone</td>
        <td>@e.Contact.Address</td>
    </tr>
}
```


Project: Run the application

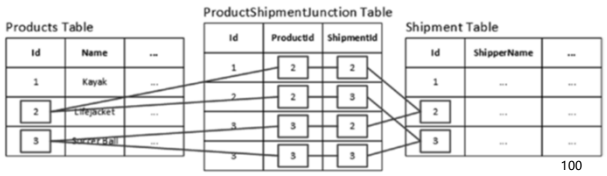
Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country	Phone	Address
1	Adam	manager	UK	123	abc
2	Cathy	sale	UK	789	ghi
3	Bob	president	CN	456	def

Create Test Data | Insert

Many-to-Many Relationship

- Entity Framework Core can represent a many-to-many relationship only by combining two one-to-many relationships and using a junction class to join them together.



Project: Update the model class

- Add a junction class.

```
public class Junction
{
    public int Id { get; set; }
    public int EmployeeId { get; set; }
    public Employee Employee { get; set; }
    public int ProjectId { get; set; }
    public Project Project { get; set; }
}
```

Project: Update the model class

```
public class Employee
{
    public int EmployeeId { get; set; }
    public string EmployeeName { get; set; }
    public string Title { get; set; }
    public int CountryID { get; set; }
    public Country Country { get; set; }
    public Contact Contact { get; set; }
    public IEnumerable<Junction> Junctions { get; set; }
}
```

```
public class Project
{
    public int ProjectId { get; set; }
    public string ProjectName { get; set; }
    public IEnumerable<Junction> Junctions { get; set; }
}
```

102

Project: Update the repository

```
public List<Employee> SelectAll()
{
    var Employees = context.Employees.
                    Include(e => e.Country).
                    Include(e => e.Contact).
                    Include(e=>e.Junctions).
                    ThenInclude(j=>j.Project);
    return Employees.ToList();
}
```

103

Project: Run the application

Employee Info List

Employee ID	Employee Name	Employee Title	Employee Country	Phone	Address	Project
1	Adam	manager	UK	123	abc	p2, p1
2	Cathy	sale	UK	789	ghi	p2
3	Bob	president	CN	456	def	p2

Create Test Data | Insert

104

THANK YOU