Saving Data to the Database

- The Add and AddRange method, which is called on the context's DbSet<T> property, makes Entity Framework Core write new data to the database.
- The SaveChanges method saves outstanding changes made to the Employee objects that are being managed by Entity Framework Core to the database.

```
context.Employees.Add(newEmployee);
context.SaveChanges();
```

58

Project: Add some test data (一)

- Update the repository

```
public interface ICountryRepository
{
    List<Country> SelectAll();
    void CreateTestData();
}
```

```
public void CreateTestData()
{
    //删除数据表所有数据
    context.Countries.RemoveRange(context.Countries);
    //添加测试数据
    context.Countries.AddRange(
        new Country[]
        {
            new Country{CountryName="USA"},
            new Country{CountryName="UK"},
            new Country{CountryName="CN"}
        }
    );
    context.SaveChanges();
}
```

59

Project: Add some test data (二)

- Update the repository

```
public interface IEmployeeRepository
{
    List<Employee> SelectAll();
    void CreateTestData();
}
```

```
public void CreateTestData()
{
    //删除数据表所有数据
    context.Employees.RemoveRange(context.Em
    //添加测试数据
    context.Employees.AddRange(
        new Employee[]
        {
            new Employee
            {
                EmployeeName="Adam",
                Title="manager",
                Country="USA"
            },
            new Employee
            {
                EmployeeName="Bob",
                Title="president",
                Country="UK"
            },
            new Employee
            {
                EmployeeName="Cathy",
                Title="sale",
                Country="USA"
            }
        }
    );
    context.SaveChanges();
}
```

60

Project: Add some test data (三)

- Update the controller

```
private ICountryRepository crepo;
public EmployeeManagerController(IEmployeeRepository r,ICountryRepository cr)
{
    repo = r;
    crepo = cr;
}
```

```
public IActionResult CreateTestData()
{
    repo.CreateTestData();
    crepo.CreateTestData();
    return RedirectToAction("List");
}
```

61

Project: Add some test data (三)

- Update the List view

```
<form>
    <button asp-action="CreateTestData">Create Test Data</button>
</form>
```

62

Project: Implement employee insertion (一)

- Update the employee repository

```
void Insert(Employee emp);
```

```
public void Insert(Employee emp)
{
    context.Employees.Add(emp);
    context.SaveChanges();
}
```

63

Project: Implement employee insertion (二)

- Update the controller

```
public IActionResult Insert(Employee model)
```

```
<form>
    <button asp-action="Insert">Insert</button>
</form>
```

64

Project: Implement employee insertion (三)

- Add an insert view

```
@model Employee
<h2>Insert New Employee</h2>
<form asp-controller="EmployeeManager" asp-action="Insert" method="post">
    <table>
        <tr>
            <td><label>Name:</label></td>
            <td><input type="text" asp-for="EmployeeName" /></td>
        </tr>
        <tr>
            <td><label>Title:</label></td>
            <td><input type="text" asp-for="Title" /></td>
        </tr>
        <tr>
            <td><label>Country:</label></td>
            <td><input type="text" asp-for="Country" /></td>
        </tr>
    </table>
    <button type="submit">Save</button>
</form>
```

65

Project: Run the application

- Can we select the country from a list?

66

Project: Make the country to select from a list (一)

- Update the insert action method

```
public IActionResult Insert()
{
    List<SelectListItem> countries =
        (from c in crepo.SelectAll()
            orderby c.CountryName ascending
            select new SelectListItem()
            {
                Text = c.CountryName,
                Value = c.CountryName
            }).ToList();
    ViewBag.Countries = countries;
    return View();
}
```

67

Project: Make the country to select from a list (二)

- Update the insert view

```
<td>
    <select asp-for="Country" asp-items="@ViewBag.Countries">
        <option value="">Please select</option>
    </select>
</td>
```

68

Project: Run the application

69

View Bag

- Action methods provide views with data to display with a view model, but sometimes additional information is required.
- Action methods can use the view bag to provide a view with extra data.

```
ViewBag.Message = "Welcome to our website";
```

70

Updating a Model with Changes

- Updating an entity typically follows the steps:
  1. Read the entity from the database.
  2. Modify the entity's properties.
  3. Save the changes to the database.

```
context.Employees.Update(changedEmployee);
context.SaveChanges();
```

```
context.Employees.Remove(e);
context.SaveChanges();
```

71

4. LINQ

What is LINQ?

- Different data source may require different access technologies.
  - Relation database: SQL
  - XML: Xpath
  - File:fread/fwrite method
- Language Integrated Query is designed to fill the gap that exists between traditional .NET languages and query languages
  - .NET language: strong typing, object-oriented
  - Query language: SQL, designed for query operations

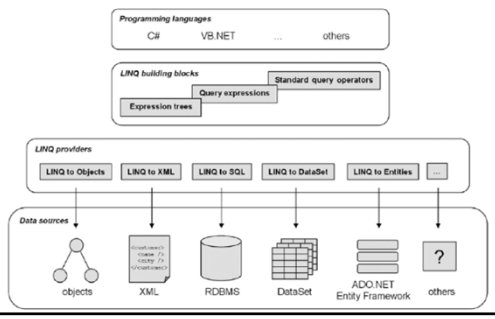
73

What Makes LINQ? (一)

- LINQ has several parts:
  - Extension methods (required)
    - Provide the functionality of LINQ
    - Such as Where , OrderBy , and Select .
  - LINQ providers (required)
    - These providers are what execute LINQ expressions in a way specific to different types of data.
    - Such as LINQ to Objects, LINQ to Entities, LINQ to XML.

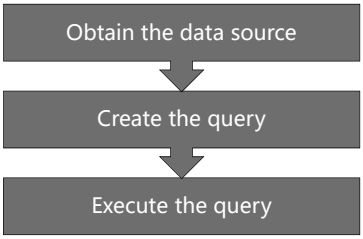
74

What Makes LINQ? (二)



75

How to Use LINQ?



76

Step 1: Obtain the Data Source

- All LINQ queries operate on objects that implement the IEnumerable<T> or IQueryable<T> interface.

```
string[] numbers = { "0042", "010", "9", "27" };
```

```
Northwind db = new Northwind(@"Data
Source=.\SQLEXPRESS;Initial
Catalog=Northwind");
```

77

Step 2: Create the Query

- LINQ queries in C# can be written by using two different syntaxes:
  - Query syntax
  - Method syntax

```
var query = from m in movies
            orderby m.Title descending;
```

```
IEnumerable<Movie> m = movies.OrderBy(m => m.Title);
```

78

Step 3: Execute the Query

- Calling most of these extension methods does not execute the query and get the results.
- Most of these extension methods return a LINQ expression that represents a question, not an answer.
- To execute the query, you must materialize it:
  - Call one of the "To" methods like ToArray or ToLookup
  - Enumerate the query
- This is called deferred execution.

79

Example of LINQ (一)

- Define the query.

```
// a string array is a sequence that implements IEnumerable<string>
string[] names = new[] { "Michael", "Pam", "Jim", "Dwight",
  "Angela", "Kevin", "Toby", "Creed" };

WriteLine("Deferred execution");

// Question: Which names end with an M?
// (written using a LINQ extension method)
var query1 = names.Where(name => name.EndsWith("m"));

// Question: Which names end with an M?
// (written using LINQ query comprehension syntax)
var query2 = from name in names where name.EndsWith("m") select name;
```

80

Example of LINQ (二)

- Execute the query.

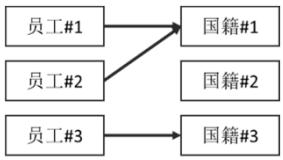
```
// Answer returned as an array of strings containing Pam and Jim
string[] result1 = query1.ToArray();

// Answer returned as a list of strings containing Pam and Jim
List<string> result2 = query2.ToList();

// Answer returned as we enumerate over the results
foreach (string name in query1)
{
  WriteLine(name); // outputs Pam
  names[2] = "Jimmy"; // change Jim to Jimmy
  // on the second iteration Jimmy does not end with an M
}
```

81

5. Data Relationship

Project: Analyze the relations between employee and country data

- There's a one-to-many relationship between the country and employee.



83

Relations Between Data

- The foundation of Entity Framework Core is the way that it represents instances of .NET classes as rows in a relational database table.
- How to define the relationship?
  - Add a navigation property in entity class.

84

Navigation Property

- Navigation property allows navigation from one object to another.
- Entity Framework Core assumes that you want to create a one-to-many relationship.
- The navigation property has been added to the class at the "many" end of the relationship.

85

Project: Add navigation property and foreign key property to Employee class

- Foreign key helps protect the integrity of the database by ensuring that a row in the Employees table can only have a value for the CountryId column that corresponds to a valid row in the Country.

```
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
}
```

86

Query the Related Data

- The Include extension method is called on objects to include related data.

87

Project: Display the related data in List view

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

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

88

Project: Run the application

Employee ID	Employee Name	Employee Title	Employee Country
1	Adam	manager	UK
2	Cathy	sale	UK
3	Bob	president	CN

89