Ordering, Filtering and Paging Data

Gill Cleeren
CTO Xebia Microsoft Services Belgium

@gillcleeren

Overview

Understanding IQueryable

Sorting and paging the overview

Searching pies

Understanding IQueryable

Important Interfaces

IEnumerable

IQueryable

```
List<int> sampleList = new List<int> { 1, 2, 3, 4, 5 };
var selection = from o in sampleList select o;
```

Working with IEnumerable

“selection” will be IEnumerable<int> here

```
namespace System.Collections.Generic
{
    public interface IEnumerable<out T> : IEnumerable
    {
        IEnumerator<T> GetEnumerator();
    }
}
```

Working with IEnumerable

Interface in System.Collection.Generic
Allows for iterating over the items in the collection
Works with simple and complex types

```
List<Pie> pies = new List<Pie>() { new Pie (){…});

IEnumerable<Pie> pieSelection = from p in pies
                                where p.IsPieOfTheWeek == true
                                select p;

foreach (Pie pie in pieSelection)
{
    Console.WriteLine($"{pie.Name}: {pie.Price}");
}
```

Using IEnumerable<T> for Complex Types

Moving to IQueryable and IQueryable<T>

Interfaces in System.Linq

IQueryable

```
namespace System.Linq
{
 public interface IQueryable :
     IEnumerable
 {
     Type ElementType { get; }
     Expression Expression { get; }
     IQueryProvider Provider { get; }
 }
}
```

IQueryable<T>

```
namespace System.Linq
{
 public interface IQueryable<out T> :
  IEnumerable<T>,
  IEnumerable,
  IQueryable
 {
 }
}
```

```
IQueryable<Pie> pies = _bethanysPieShopDbContext.Pies.AsQueryable();

var pies = from p in _bethanysPieShopDbContext.Pies
           select p;
```

Using IQueryable<T>

Using IEnumerable

Fetching records without filtering

```
IEnumerable<Pie> pieSelectionEnumerable =
    _bethanysPieShopDbContext.Pies.Where(p => p.IsPieOfTheWeek);

pieSelectionEnumerable = pieSelectionEnumerable.Take(2);

foreach (Pie pie in pieSelectionEnumerable)
{
  Console.WriteLine($"{pie.Name}: {pie.Price}");
}
```

The Resulting Query on the Database

```
      SELECT [p].[PieId], [p].[AllergyInformation], [p].[CategoryId], [p].[ImageThumb
nailUrl], [p].[ImageUrl], [p].[InStock], [p].[IsPieOfTheWeek], [p].[LongDescription],
 [p].[Name], [p].[Price], [p].[RowVersion], [p].[ShortDescription]
      FROM [Pies] AS [p]
      WHERE [p].[IsPieOfTheWeek] = CAST(1 AS bit)
Caramel Popcorn Cheese Cake: 22,95
Chocolate Cheese Cake: 19,95
```

Using IQueryable

Fetching records without filtering

```
IQueryable<Pie> pieSelectionQueryable =
    _bethanysPieShopDbContext.Pies.Where(p => p.IsPieOfTheWeek).AsQueryable();

pieSelectionQueryable = pieSelectionQueryable.Take(2);
foreach (Pie pie in pieSelectionQueryable)
{
    Console.WriteLine($"{pie.Name}: {pie.Price}");
}
```

The Resulting Query on the Database

```
info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (34ms) [Parameters=[@__p_0='?' (DbType = Int32)], CommandTyp
e='Text', CommandTimeout='30']
      SELECT TOP(@__p_0) [p].[PieId], [p].[AllergyInformation], [p].[CategoryId], [p]
.[ImageThumbnailUrl], [p].[ImageUrl], [p].[InStock], [p].[IsPieOfTheWeek], [p].[LongD
escription], [p].[Name], [p].[Price], [p].[RowVersion], [p].[ShortDescription]
      FROM [Pies] AS [p]
      WHERE [p].[IsPieOfTheWeek] = CAST(1 AS bit)
Caramel Popcorn Cheese Cake: 22,95
Chocolate Cheese Cake: 19,95
```

```
await
     _bethanysPieShopDbContext.Categories
     .AsNoTracking()
     .OrderBy(c => c.CategoryId)
     .ToListAsync();
```

Understanding Deferred Execution

Only when the results are iterated, will the query run

Sorting and Paging the Overview

Demo

Adding the PaginatedList

Extending the repository

Using paging on the view

Demo

Adding sorting on the column headers

Searching pies

Demo

Creating the search filter

Summary

IQueryable is used to query the database

Deferred execution allows for optimal time to perform the querying

Up Next:

Adding More Complex Features to the Site