CIS 365 Lab08: Collections and Language Integrated Query

Adam Lewis

May 25, 2018

Contents

1 Objectives

1. Gain an understanding of the **System.Collections** frameworks, particularly the **List** collection

2. Introduce C#'s Language Integrated Query (LINQ) feature

2 Background

It is common for students to take CS372 (Data Structures) at the same time as they take this class. Students in CS372 experience the unmitigated joy of having to implement your own set of classes to work with the data structures

discussed in that course. Now that you've done that once, you can start to take advantage of the fact that language and platform providers provide implementations of these classes.

2.1 System.Collection

The .NET Framework Class Library provides a set of "Collection" classes that implement many of the common data structures required by applications. As this is provided as part of .NET, these classes are available to any language that works with the .NET Common Language Runtime (Visual C++, Visual BASIC, and, of course, C#).

The Collection classes are built on top of C# *generics* feature; that means that you create a collection of some generic type that you parameterize at run-time. For example, a list of integers can be declared as:

```
List<int> aListOfIntegers;
List<string> aListofStrings;
```

All of the Collection classes support a common message protocol. Methods in this protocol include:

Method	**Description**
Add	Add an element to the collection
Clear	Removes all elements from the collection
Contains	Returns **true** if the list contains the specified element
Count	Property that stores number of elements in the collection
Insert	Inserts an element at the specified index in the list
Remove	Removes the first occurence of teh specified value
RemoveAt	Removes the element at the specified index
RemoveRange	Removes a specified number of elements starting at index
Sort	Sorts the collection
IndexOf	Returns the index of the first occurrence of a value
Capacity	Property that says how many elements can be stored without resizing
TrimeExcess	Set the capacity to the number of elements collection contains

2.2 Language Integrated Query (LINQ)

So many applications use databases and similar application servers that the designers of C# included a feature known as "Language Integrated Query"

(LINQ). Prior to this feature being added to the language, application developers were required to embed SQL statements in their programs, call a function to process those statements, and write code to parse the returned records into a form that the program could understand. LINQ moves this functionality directly in the language so that you can write query expressions similar to SQL statements that retrieve information from a variety of sources, not just databases.

LINQ operates on the concept of a "LINQ Provider" which a set of classes that implement LINQ operations and the connection to the underlying service. In this lab, we will be using the "LINQ to Objects" provider to perform LINQ queries on **System.Collection.List** objects.

3 Instructions

This lab will use LINQ to query for Employee data stored in a list of objects.

3.1 Setup

A C# source code file containing the Employee class is attached to the Lab assignment in Blackboard. Start Visual Studio and create a new Console application named "LinqDemo". Download the file *employee.cs* to your project directory. Select the project in the Solution Explorer, right click, and select *Add-Existing Item*. Follow the prompts in the dialog box to add the file.

In your main class file, make certain to include the **System.Collections.Generic** and **System.Linq** frameworks.

3.2 Data initialization

Add the following code to the **Main()** function:

```
var employees = new List<Employee>
{
  new Employee( "Jason", "Red", 5000M ),
  new Employee( "Ashley", "Green", 7600M ),
  new Employee( "Matthew", "Indigo", 3587.5M ),
  new Employee( "James", "Indigo", 4700.77M ),
  new Employee( "Luke", "Indigo", 6200M ),
  new Employee( "Jason", "Blue", 3200M ),
  new Employee( "Wendy", "Brown", 4236.4M )
}; // end init list
```

```
// display all employees
Console.WriteLine( "Original list:" );
foreach ( var element in employees )
    Console.WriteLine( element );
```

We now have a list of objects to search.

3.3 Some simple queries

Now lets write a LINQ query that will filter the list for a range of salaries:

```
var between4k6k =
    from e in employees
    where e.MonthlySalary >= 4000M && e.MonthlySalary <= 6000M
    select e;
      Console.WriteLine( string.Format(
      "\nEmployees earning in the range {0:C}-{1:C} per month:",
      4000, 6000 ) );

foreach ( var element in between4K6K )
      Console.WriteLine( element );
```

A similar query that will sort the data by last name and then first name:

```
// order the employees by last name, then first name with LINQ
var nameSorted =
    from e in employees
    orderby e.LastName, e.FirstName
    select e;

// header
Console.WriteLine( "\nFirst employee when sorted by name:" );

// attempt to display the first result of the above LINQ query
if ( nameSorted.Any() )
   Console.WriteLine( nameSorted.First() );
else
   Console.WriteLine( "not found" );
```

3.3.1 Your turn

Use LINQ to write a query that will display a sorted list of the full names of the employees.

4 Submission instructions

Combine a copy of your source code and a screen-shot of your program into a PDF file (you can use Microsoft Word to do this) and attach it to the assignment on Blackboard. Emacs 24.4.1 (Org mode 8.2.10)