Tutorial 8: Data Sources and LINQ

Purpose: To practise the use of LINQ for filtering and searching data structures, and to communicate with the RAP case study's MySQL database.

Outline

The Language Integrated Query (LINQ) feature of C# (and also Visual Basic) allows common collection processing tasks to be written in a consistent form within the host language while being applicable to a variety different data sources: arrays, Collection types, XML documents and databases (Microsoft's SQL Server natively and others with suitable third-party libraries such as LinqConnect [this is not an endorsement of that particular product, merely an example]).

In the absence of a suitable third-party library, communicating with a MySQL database requires that we fall back on classes for issuing native SQL statements expressed as strings and the general-purpose and hence awkward-to-use MySqlDataReader and MySqlDataAdapter classes to read the results of queries. In the future you will often use technologies like Entity Framework (with .NET) and Hibernate (with Java) to map between your system's classes and database tables, but whenever you are faced with a legacy database system you will need to fall back on the approach described here.

The following tasks are based on the last two tutorials. Either load your project from the last tutorial or download the sample solution as a starting point.

1 LINQ to Objects

C# provides considerable flexibility in querying data structures. In addition to the traditional approach of writing a `for` or `foreach` loop, collection objects may be queried using short lambda expressions (a topic we do not cover due to time constraints), or by using LINQ. LINQ queries are typically longer than the equivalent operation using a lambda expression, but they are also more easily parsed by a human reader.

1.1 Filtering with LINQ

In the previous C# tutorials you filtered a list of Employee objects by instantiating a new list and using a `foreach` loop to iterate over a master list and to select the objects matching some condition (previously it was Employee gender). Locate the Filter() method where you wrote that code and replace it with the following, which you are to complete:

```
var filtered = Some LINQ expression
return new List<Employee>(filtered);
```

Task: Replace *Some LINQ expression* with an actual LINQ expression that selects Employee objects whose gender matches the value passed to the method. Your tutor will provide assistance and, once everyone's had an opportunity to attempt this, go through a possible solution with the group.

1.2 Assignment experimentation: Who's been trained?

This task may be left until after the tutorial if you believe you will not be able to get much of Part 2 done before the end. Note, however, that task 2.3 requires that the following is completed first.

In this task you will treat publications in the database as if they are training sessions that an employee can complete, where a publication's *available* date will be treated as the date an employee was certified for that training session. Create a new class called **TrainingSession** that contains auto-

generated properties called Title (a string), Year (an int), Certified (a DateTime object) and Mode (which will be of type Mode, a new enum type you will define). In the same .cs file, outside the definition of the TrainingSession class, add the following declaration for the Mode enum type:

```
public enum Mode { Conference, Journal, Other };
```

Add a ToString() method that returns a string that combines the Title, Certified date and Mode in some way, such as *The Title* completed by *Mode* on *Certified*.

Optionally, add another, read-only property with the following structure:

```
public int Freshness
{
    get { return some expression you are to write; }
}
```

Using any resources available to you (MSDN, IntelliSense [the pop-up help the editor provides], etc.) replace the grey text with an expression that subtracts the Certified date from today's date and then evaluates the number of days that difference represents.

1.2.1 Create some TrainingSessions

For testing purposes, instantiate and populate a List of TrainingSessions in Main(). Include at least five objects, making sure that they occur over a range of years between 2012 and 2017.

1.2.2 Filter the list of TrainingSessions

Create a LINQ expression that finds all TrainingSessions that have a Freshness of no more than 200 days. **Make sure you add some code to display the filtered list to check that it works.**

Create another LINQ expression that finds all the TrainingSessions that occurred between 2012 and 2016 (inclusive). When you're satisfied both work, consider commenting out the testing code before moving to Task 2.

2 Communicating with a MySQL database

The tasks in this section rely on the following database connection information:

Database, User Id, and Password: kit206 **Data Source:** alacritas.cis.utas.edu.au

Refer to the Module 5 presentation slides and accompanying SqlCommandDemo.cs file for the relevant code for establishing a connection to a MySQL database, issuing a basic query, and reading the result. The remainder of the tutorial will assume you have these materials available.

2.1 Connecting to the Employment & Research Database

Task: Modify the Agency class so that it has a (static) MySqlConnection field called `conn`. Add this method to Agency, which will allow different methods within it to obtain a reference to `conn` when needed. You won't need to have as much line wrapping in your code.

```
private static MySqlConnection GetConnection() {
   if (conn == null) {
      string connectionString =
        String.Format("Database={0};Data Source={1};User Id={2}; Password={3}",
        db, server, user, pass);

      conn = new MySqlConnection(connectionString);
   }
   return conn;
}
```

Note that, in your assignment, if your bridge to the database is implemented by an object, and not by class methods in an abstract class, then code like this would go in the constructor (and the check if conn is `null` could be omitted).

2.2 Reading Researchers into Employee

You will have recognised that the Employee class developed in recent tutorials is similar to whichever class you have in your assignment to represent a researcher and its subtypes staff and students. In fact, because we are taking complete control over the mapping from database to application object, there's no reason why we can't treat it that way as part of this learning exercise.

Task: Create a new method in Agency called LoadAll(), with the same return type as the existing Generate() method. Using SqlCommandDemo.ReadData() as a guide, implement this method so that it selects the **id**, **given_name** and **family_name** columns from the **researcher** table. The relevant changes are:

- Near the start of the method call `GetConnection()` to ensure that `conn` exists; you may assign the result to a local MySqlConnection variable if you wish
- Modify the SQL select statement string
- Declare a new List of Employee objects
- Each time through the loop over the result set, instead of writing the retrieved data to the console, create a new Employee, setting:
    - its ID to `rdr.GetInt32(0)` (assuming you selected **id** first)
    - its Name to the concatenation of `rdr.GetString(1)` and `rdr.GetString(2)`

Remember to return the new list of Employee objects at the end.

Modify Boss so that it calls LoadAll() instead of Generate() and run the program to test.

2.3 What 'training' has an employee done?

Add a new method to Agency called LoadTrainingSessions() that will query the database for all publications by a given Employee (i.e., researcher) ID passed as a parameter and return a List of TrainingSessions. The *title* and *year* will map to the same properties in the TrainingSession class, while a publication's *type* will be loaded into Mode and its *available* date will be loaded into Certified.

2.3.1 Constructing the query

Follow the pattern from 2.2, but modify your query code to be the following:

```
MySqlCommand cmd = new MySqlCommand("select title, year, type, available " +
            "from publication as pub, researcher_publication as respub " +
            "where pub.doi = respub.doi and researcher_id=?id", conn);

cmd.Parameters.AddWithValue("id", id);
```

What does the ?id mean? Although we could have simply appended the integer id the method accepted as a parameter to the query string, it is generally better to separate the unchanging parts of a query from those parts that will change. When communicating with a MySQL database ?*name* is a template parameter with the identifier *name*.

2.3.2 Creating the TrainingSessions

Given the query above, and a MySqlDataReader called rdr, the Title, Year and Certified properties of TrainingSession can be set to `rdr.GetString(0)`, `rdr.GetInt32(1)` and `rdr.GetDateTime(3)`, respectively, as in the following code:

```
new TrainingSession{ Title=rdr.GetString(0), Year=rdr.GeInt32(1),
                                          Certified=rdr.GetDateTime(3) };
```

To do a quick test that this works, modify Main() to call Agency.LoadTrainingSessions() with a researcher ID you know is in the database and then display each one on the console.

Alternatively (and optionally), if you're feeling confident and have time right now, add a List<TrainingSession> property to Employee called Skills and add a method to Boss to load the List of TrainingSessions for each Employee in turn, then modify the Employee ToString() method to include each of its TrainingSessions. Although this takes longer, it makes one of the later tasks easier.

Whichever approach you chose, test your program to view some loaded TrainingSessions.

2.3.3 Converting strings to Enums

Currently the loaded TrainingSessions have a title, year and certification date, but no mode (i.e., type). As we have defined an enum type Mode with the same values as appear in the database column *publication.type*, surely it should be easy to convert the string value in the database to the equivalent Enum value.

Oddly, whereas C# allows many things to be expressed more succinctly than in Java, converting a string to an Enum is ridiculously complicated in C# (and trivial in Java). So, add this general purpose method to Agency to assist in the conversion:

```
public static T ParseEnum<T>(string value)
{
   return (T)Enum.Parse(typeof(T), value);
}
```

Then, given a string s, it can be converted to a value of type Mode via:

```
Mode mode = ParseEnum<Mode>(s);
```

Modify the code that creates each TrainingSession to use the ParseEnum method. (Note that, apart from the fiddly conversion from database string value to Enum value, Enums do make the rest of your code more robust.)

2.3.4 How many training sessions has a staff member attended recently?

Now that you can load a collection of TrainingSessions for a given Employee you can determine how many sessions an Employee has attended in the last two calendar years. There are two alternative options for completing this task, both of which are valid but would be used in different situations:

Option 1: If you have added a Skills property to Employee and loaded training/publications into it, add a method to Employee called RecentTraining() that returns the number of TrainingSessions whose year is either this year or the last (do *not* write the actual year numbers into your code). Use a LINQ expression to select each Training object with the year in the correct range, then use the Count() method on the result of the LINQ expression to determine how many there are. To test this approach, modify Employee.ToString() to include the result of RecentTraining().

Option 2: Alternatively, write a new method in Agency that uses a "SELECT COUNT(*)…" query to perform the same test directly on the database. The method should accept either an Employee or an integer ID, and may optionally include the start and end years to use.

3 For your assignment

Look for similarities between today's exercises and the functions required of your assignment application. Adapt your approaches to suit the needs of the assignment.

4 Not covered: disconnected data and DataSets

The potential scope for this unit is extremely large. One of the approaches to loading data from a database into a DataSet object, is covered briefly in Module 5 but is not essential for implementing the assignment, so is not covered in this tutorial. Be aware that it is an alternative that can make some data processing tasks easier, especially when modifying the contents of a database.