Database access in C# using LINQ

Hans-Wolfgang Loidl
<H.W.Loidl@hw.ac.uk>

**School of Mathematical and Computer Sciences,
Heriot-Watt University, Edinburgh**

Semester 1 — 2021/22

HERIOT WATT UNIVERSITY

ADO.NET

- ADO.NET provides a direct interface to a database.
- The interface is database-specific.
- ADO.NET uses a conventional, shallow embedding of SQL commands into C# as host language, i.e. SQL commands are composed as strings.
- A more advanced, deep embedding of SQL commands is provided by LINQ, i.e. SQL commands a language constructs.

HERIOT WATT UNIVERSITY

Structure of database access

To access a database with ADO.NET the following steps are necessary:

- Connect to a database
- Compose an SQL query
- Issue the query
- Retrieve and process the results
- Disconnect from the database.

HERIOT WATT UNIVERSITY

ADO.NET Example

To connect to a database, a connection string has to specify location, account, password etc. (fill in user id and pwd):

```
1 using MySql.Data.MySqlClient;
2 string cstr = "Server=anubis;Database=test;User␣ID=;
    Password=";
3 MySqlConnection dbcon;
4
5 try {
6   dbcon = new MySqlConnection(cstr);
7   dbcon.Open();
8 }  catch (MySql.Data.MySqlClient.MySqlException ex) {
    ... }
```

HERIOT WATT UNIVERSITY

ADO.NET Example (cont'd)

- Next, compose an SQL query as a string
- This can be any SQL operation
- Depending on the underlying database, SQL extensions might be available.

```
1 MySqlCommand dbcmd = dbcon.CreateCommand();
2
3 string sql = "SELECT␣A_ID,␣A_FNAME,␣A_LNAME␣" +
4              "FROM␣authors";
5 dbcmd.CommandText = sql;
```

HERIOT WATT UNIVERSITY

ADO.NET Example (cont'd)

Next, issue the query, and process the result, typically in a while loop.

```
1 MySqlDataReader reader = dbcmd.ExecuteReader();
2
3 while(reader.Read()) {
4   string FirstName = (string) reader["A_FNAME"];
5   string LastName = (string) reader["A_LNAME"];
6   Console.WriteLine("Name: " + FirstName + " " +
        LastName);
7 }
```

ADO.NET Example (cont'd)

Finally, clean-up and disconnect.

```
1 reader.Close();
2 reader = null;
3 dbcmd.Dispose();
4 dbcmd = null;
5 dbcon.Close();
6 dbcon = null;
```

HERIOT WATT UNIVERSITY

LINQ

- Language Integrated Query (LINQ) is a more advanced way to interact with databases.
- It's a new feature with C# 3.0 onwards.
- It provides SQL-like commands as language extensions, rather than composing SQL queries as strings (*deep embedding*)
- It can also be used to access other forms of data, such as XML data or compound C# data structures.

HERIOT WATT UNIVERSITY

LINQ Example

- **The same example as before, written in LINQ is much simpler.**
- **First, classes, representing the tables of the database are defined.**

```
1  [Table(Name = "authors")]
2  public class Authors
3  {
4    [Column]
5    public int A_ID    { get ; set ; }
6    [Column]
7    public string A_FNAME  { get ; set ; }
8    [Column]
9    public string A_LNAME  { get ; set ; }
10 }
```

HERIOT WATT UNIVERSITY

LINQ Example (cont'd)

Next, a connection is established, using a connection string similar to ADO.NET:

```
1 DataContext db =
2     new DataContext("Data␣Source␣=␣.\\MySql;" +
3                     "Initial␣Catalog=test;Integrated␣
                        Security=True");
4
5 DataContext db = new DataContext(connStr);
```

HERIOT WATT UNIVERSITY

LINQ Example (cont'd)

The main advantage of LINQ is the simplified way of performing queries.

```
1 Table<Authors> AuthorTable = db.GetTable<Authors>();
2 List<Authors> dbQuery = from author in Authors select
      author ;
3
4 foreach (var author in dbQuery) {
5     Console.WriteLine("Author: "+author.A_FNAME+' ' '+
          author.A_LNAME);
6 }
```

Note, that SQL-like commands such as `select`, `from` etc are directly available

Querying in-memory Data

- LINQ can also be used to query in-memory data, such as XML data or compound C# data structures.
- This results in more uniform and succinct code.
- Using LINQ in this way requires several advanced language features.
- It is an alternative to using standard mechanisms of traversing data structures such as iterators.

HERIOT WATT UNIVERSITY

Example

Assume we have a list of books:

```
List<Book> booklist = new List<Book> {
          new Book { Title = "Learning␣C#"
                   , Author = "Jesse␣Liberty"
                   , Publisher = "O'Reilly"
                   , Year = 2008
                   },
          new Book { Title = "Programming␣C#"
                   , Author = "Jesse␣Liberty"
                   , Publisher = "O'Reilly"
                   , Year = 2008
                   },
          new Book { Title = "Programming␣PHP"
                   , Author = "Rasmus␣Lerdorf,␣Kevin␣
                     Tatroe"
                   , Publisher = "O'Reilly"
                   , Year = 2006
                   },

```

Example (conventional)

The conventional way to iterate over the list looks like this:

```
1 foreach (Book b in booklist) {
2     if (b.Author ==  "Jesse␣Liberty") {
3       Console.WriteLine(b.Title + "␣by␣" + b.Author);
4     }
5 }
```

Example (LINQ)

In contrast, the LINQ-style iteration looks like an SQL query and is shorter:

```
1  IEnumerable<Book> resultsAuthor =
2            from b in booklist
3            where b.Author == "Jesse␣Liberty"
4            select b;
5
6  Console.WriteLine("LINQ␣query:␣find␣by␣author␣...");
7  // process the result
8  foreach (Book r in resultsAuthor) {
9    Console.WriteLine(r.Title + "␣by␣" + r.Author);
10 }
```

Example (LINQ)

In contrast, the LINQ-style iteration looks like an SQL query and is shorter:

```
1  IEnumerable<Book> resultsAuthor =
2            from b in booklist
3            where b.Author == "Jesse␣Liberty"
4            select b;
5
6  Console.WriteLine("LINQ␣query:␣find␣by␣author␣...");
7  // process the result
8  foreach (Book r in resultsAuthor) {
9    Console.WriteLine(r.Title + "␣by␣" + r.Author);
10 }
```

Embedded SQL-like code (LINQ code)

Example (LINQ)

In contrast, the LINQ-style iteration looks like an SQL query and is shorter:

```
1  IEnumerable<Book> resultsAuthor =
2              from b in booklist
3              where b.Author == "Jesse␣Liberty"
4              select b;
5
6  Console.WriteLine("LINQ␣query:␣find␣by␣author␣...");
7  // process the result
8  foreach (Book r in resultsAuthor) {
9    Console.WriteLine(r.Title + "␣by␣" + r.Author);
10 }
```

Embedded SQL-like code (LINQ code)

Iterate over the results

Example (with anonymous types)

To avoid returning entire book results from the query we can use *anonymous types* and just return title and author:

```
1  // NB: this needs to infer the type (anonymous!)
2  var resultsAuthor1 =
3      from b in booklist
4      where b.Author == "Jesse␣Liberty"
5      // NB: anonymous type here!
6      select new { b.Title, b.Author} ;
7
8  // process the result
9  foreach (var r in resultsAuthor1) {
10   Console.WriteLine(r.Title + "␣by␣" + r.Author);
11 }
```

Example (with anonymous types)

To avoid returning entire book results from the query we can use *anonymous types* and just return title and author:

```
1  // NB: this needs to infer the type (anonymous!)
2  var resultsAuthor1 =
3      from b in booklist
4      where b.Author == "Jesse␣Liberty"
5      // NB: anonymous type here!
6      select new { b.Title, b.Author} ;
7
8  // process the result
9  foreach (var r in resultsAuthor1) {
10   Console.WriteLine(r.Title + "␣by␣" + r.Author);
11 }
```

Anonymous type (var)

HERIOT WATT UNIVERSITY

Example (with lambda expressions)

Lambda expressions **can be used to shorten the query even further:**

```
1 // NB: lambda expression here
2 var resultsAuthor2 =
3  booklist.Where(bookEval => bookEval.Author == "Jesse␣
      Liberty");
4
5 // process the result
6 foreach (var r in resultsAuthor2) {
7   Console.WriteLine(r.Title + "␣by␣" + r.Author);
8 }
```

Example (with lambda expressions)

Lambda expressions **can be used to shorten the query even further:**

```
1 // NB: lambda expression here
2 var resultsAuthor2 =
3  booklist.Where(bookEval => bookEval.Author == "Jesse␣
      Liberty");
4
5 // process the result
6 foreach (var r in resultsAuthor2) {
7   Console.WriteLine(r.Title + "␣by␣" + r.Author);
8 }
```

Lambda expression (anonymous function)

HERIOT WATT UNIVERSITY

Example (more SQL-like commands)

We can sort the result by author:

```
var resultsAuthor3 =
          from b in booklist
          orderby b.Author
          // NB: anonymous type here!
          select new { b.Title, b.Author} ;

Console.WriteLine("LINQ query: ordered by author ...")
    ;
// process the result
foreach (var r in resultsAuthor3) {
  Console.WriteLine(r.Title + " by " + r.Author);
}
```

HERIOT WATT UNIVERSITY

Example (joining tables)

We can join tables like this:

```
var resultList4 =
  from b in booklist
  join p in purchaselist on b.Title equals p.Title
  where p.Quantity >=2
  select new { b.Title, b.Author, p.Quantity } ;

Console.WriteLine("LINQ query: ordered by author ...")
    ;
// process the result
foreach (var r in resultList4) {
  Console.WriteLine(r.Quantity + " items of " + r.
      Title
                        + " by " + r.Author);
}
```

HERIOT WATT UNIVERSITY

Summary

- C# supports two ways of querying databases:
  - ADO.NET with SQL queries as strings
  - LINQ with SQL commands embedded into the language
- ADO.NET is older and more robust
- LINQ is newer and easier to use
- LINQ can also be used to traverse in-memory data structures.

HERIOT WATT UNIVERSITY