UNIVERSITY OF WESTMINSTER

Enterprise Application Development

Lecture 8 – Working with Databases (in C# and .NET)

UNIVERSITY OF WESTMINSTER

What is a database?

- A collection of structured information

- Database software is used to create, edit, and maintain database files and records
  - SQL (Structured Query Language)

- Alternatives to databases
  - Text files
  - Spreadsheets

Database types

- Relational
  - Items in a relational database are organized as a set of tables with columns and rows
  - Relational database technology provides the most efficient and flexible way to access structured information

- Non-relational (NoSQL)
  - Allows unstructured and semi-structured data to be stored and manipulated
  - NoSQL databases grew popular as web applications became more common and more complex

UNIVERSITY OF WESTMINSTER

Database operations

- **Create table**
  - CREATE TABLE Person (Id int, Name varchar, Surname varchar, Age int);
- **Insert**
  - INSERT INTO Person (Name, Surname, Age) VALUES (“John”, “Doe”, 21);
- **Update**
  - UPDATE Person SET Age=35 WHERE Name=“John”;
- **Delete**
  - DELETE FROM Person WHERE Age=25;

Database operations

- **Queries**
  - **Where**
    - SELECT * FROM Person WHERE Age < 30;
  - **Select**
    - SELECT Name, Age FROM Person;
  - **Order by**
    - SELECT * FROM Person ORDER BY Age;
  - **Join**
    - SELECT Name, Surname, Age FROM Person JOIN Student ON Person.Id = Student.Id;
  - **Group by**
    - SELECT Age, COUNT(*) FROM Person GROUP BY Age;

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();

            builder.DataSource = "<your_server.database.windows.net>";
            builder.UserID = "<your_username>";
            builder.Password = "<your_password>";
            builder.InitialCatalog = "<your_database>";

            using (SqlConnection connection = new SqlConnection(builder.ConnectionString))
            {
                Console.WriteLine("\nQuery data example:");
                Console.WriteLine("=========================================\n");

                connection.Open();

                String sql = "SELECT name, collation_name FROM sys.databases";

                using (SqlCommand command = new SqlCommand(sql, connection))
                {
                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            Console.WriteLine("{0} {1}", reader.GetString(0), reader.GetString(1));
                        }
                    }
                }
            }
        }
        catch (SqlException e)
        {
            Console.WriteLine(e.ToString());
        }
    }
}
```

UNIVERSITY OF WESTMINSTER

- Prints
  - Database name
  - Character set

UNIVERSITY OF WESTMINSTER

ORM

- ORM (Object-Relational Mapper) allows us to connect a programming language to the database system and work with objects instead of string queries.

- Benefits
  - No string queries which means the code is statically typed
  - No need to learn a database query language (SQL)
  - Makes the application independent of the database management system
  - Easier to (unit) test the code

UNIVERSITY OF WESTMINSTER

Entity Framework

- EF – Entity Framework is an Object-Relational Mapper library
- Enables .NET developers to work with a database using .NET objects
- Eliminates the need for most of the data-access code that typically needs to be written
- EF is very flexible and supports different development approaches

Entity Framework

- Model development approaches
  - Generate a model from an existing database
  - Hand code a model to match the database
  - Create a database from the model

UNIVERSITY OF WESTMINSTER

Entity Framework

- EF Core supports many database engines
  - SQL Server
  - SQLite
  - In Memory DB
  - Cosmos DB
  - PostgreSQL
  - MySQL/MariaDB
  - Google Spanner
  - … (https://docs.microsoft.com/en-us/ef/core/providers/?tabs=vs)

Entity Framework

- DbContext
  - The context object allows querying and saving data
  - We need a class that inherits from DbContext and an instance of this class will represent a session with the database
- DbSets
  - Models that will be used to represent tables and describe data scheme
- Migrations
  - Allow evolving the database as the model changes

Entity Framework

- Example

```csharp
public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }
    public int Rating { get; set; }
    public List<Post> Posts { get; set; }
}

public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }

    public int BlogId { get; set; }
    public Blog Blog { get; set; }
}
```

```csharp
public class BloggingContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(
            @"Server=(localdb)\mssqllocaldb;Database=Blogging;Trusted_Connection=True");
    }
}
```

UNIVERSITY OF WESTMINSTER

Entity Framework

- Save data

```
using (var db = new BloggingContext())
{
    var blog = new Blog { Url = "http://sample.com" };
    db.Blogs.Add(blog);
    db.SaveChanges();
}
```

- Query data

```
using (var db = new BloggingContext())
{
    var blogs = db.Blogs
        .Where(b => b.Rating > 3)
        .OrderBy(b => b.Url)
        .ToList();
}
```

UNIVERSITY OF WESTMINSTER

LINQ

- LINQ – Language Integrated Queries is a set of technologies based on the integration of query capabilities directly into the C# language
- Can be used for querying in memory data structures or databases
- Traditionally, queries against data are expressed as simple strings without type checking at compile time or IntelliSense support

UNIVERSITY OF WESTMINSTER

LINQ - Syntax types

- Query

```
int[] numbers = { 5, 10, 8, 3, 6, 12};

//Query syntax:
var numQuery1 =
    from num in numbers
    where num % 2 == 0
    orderby num
    select num;

PrintResult(numQuery1);
```

- Method

```
//Method syntax:
var numQuery2 =
numbers.Where(num => num % 2 == 0)
.OrderBy(n => n);

PrintResult(numQuery2);
```

```
/*
    Output:
    6 8 10 12
    6 8 10 12
 */
```

UNIVERSITY OF WESTMINSTER

LINQ - Operations

- Obtaining a data source

```
//queryAllCustomers is an IEnumerable<Customer>
var queryAllCustomers = from cust in customers
                        select cust;
```

- Filtering

```
var queryLondonCustomers = from cust in customers
                           where cust.City == "London"
                           select cust;
```

```
where cust.City == "London" && cust.Name == "Devon"
```

LINQ - Operations

- Ordering (A to Z)

```
var queryLondonCustomers3 =
    from cust in customers
    where cust.City == "London"
    orderby cust.Name ascending
    select cust;
```

- Ordering (Z to A)

```
var queryLondonCustomers3 =
    from cust in customers
    where cust.City == "London"
    orderby cust.Name descending
    select cust;
```

LINQ - Operations

- Grouping

```csharp
// queryCustomersByCity is an IEnumerable<IGrouping<string, Customer>>
  var queryCustomersByCity =
      from cust in customers
      group cust by cust.City;

  // customerGroup is an IGrouping<string, Customer>
  foreach (var customerGroup in queryCustomersByCity)
  {
      Console.WriteLine(customerGroup.Key);
      foreach (Customer customer in customerGroup)
      {
          Console.WriteLine(customer.Name);
      }
  }
```

UNIVERSITY OF WESTMINSTER

LINQ - Operations

- Selecting single property

```
var countryNames = countryList.Where(c => c.Language=="English")
.Select(s => s.Name);
```

- Selecting multiple properties

```
var cDetails = from c in countryList
where c.Language != "English"
select new { CountryName = c.Name, Language= c.Language };
```

LINQ - Operations

- Joining

```
var innerJoinQuery =
    from cust in customers
    join dist in distributors on cust.City equals dist.City
    select new { CustomerName = cust.Name, DistributorName = dist.Name };
```

```
foreach (var item in innerJoinQuery)
{
    Console.WriteLine("Customer name {0}, Distributor name {1}",
       item.CustomerName, item.DistributorName);
}
```

UNIVERSITY OF WESTMINSTER

NuGet and EF

- NuGet is a (.NET) package manager, and it enables developers to share reusable code
- Add package to get access to DbContext: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore/
- Add package to use UseSqlServer: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.SqlServer/
- Add package to work with Migrations: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.Tools/

Resources

- Useful links
  - https://docs.microsoft.com/en-us/ef/core/
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries
- Books
  - Databases Illuminated, 3rd Edition, Catherine M. Ricardo & Susan D. Urban
  - Entity Framework Core in Action, Second Edition, Jon P Smith