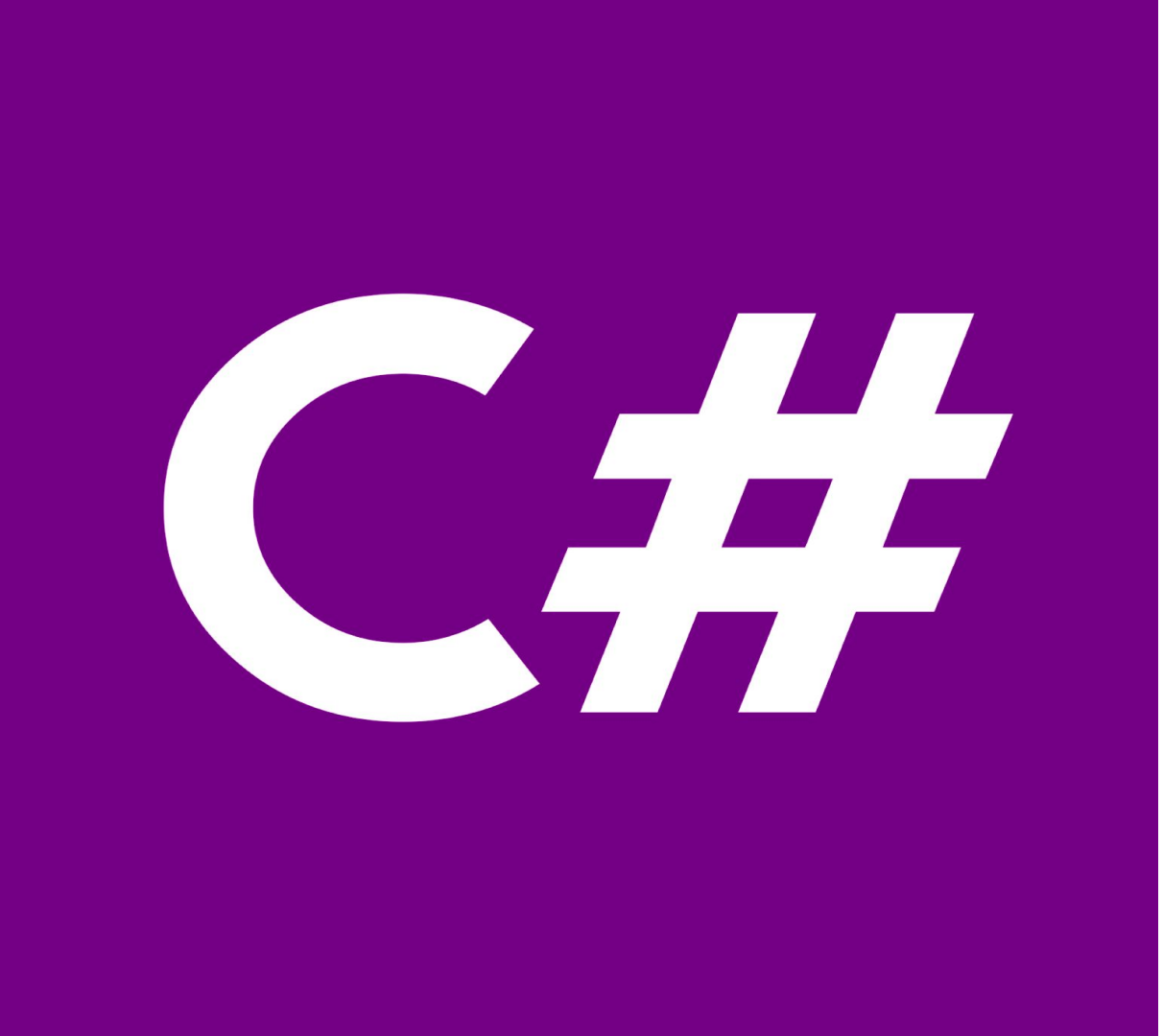
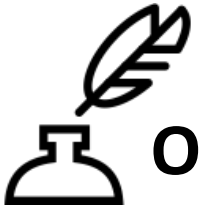


TDDD49/725G66
C# and .NET Programming

(Lecture 04)

Sahand Sadjadee
Department of Information and Computer Science
Linköping University



Outline

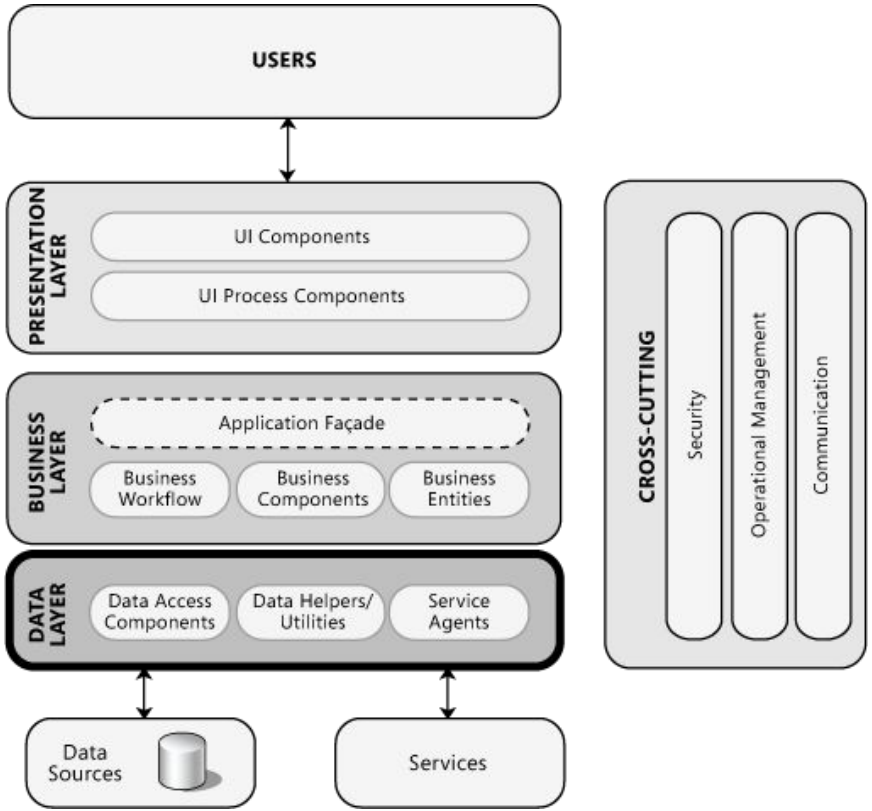
1. The Data Access Layer(DAL)
2. LINQ
3. Repository Pattern

LINKÖPINGS UNIVERSITET

The Data Access Layer

The Data Access Layer

https://msdn.microsoft.com/en-us/library/ee658127.aspx



- **Data Access components**. These components abstract the logic required to access the underlying data stores. They centralize common data access functionality in order to make the application easier to configure and maintain.
- **Service agents**. When a business component must access data provided by an external service, you might need to implement code to manage the semantics of communicating with that particular service.

LINKÖPINGS UNIVERSITET

Storage Options

- Local file (JSON/XML)
- Embedded databases (SQLite)
- DataBase Managements Systems (Oracle, MySQL, Microsoft SQL Server, MongoDB and …)
- Service-oriented Storage (Firebase Storage)
  http://wikibon.org/wiki/v/Service-oriented_storage:_An_idea_whose_time_has_come
  http://www.pcworld.com/article/2919372/google-launches-a-service-for-storing-big-data.html

LINKÖPINGS UNIVERSITET

General Design Considerations

https://msdn.microsoft.com/en-us/library/ee658127.aspx

- Choose an appropriate data access technology.
- Use abstraction to implement a loosely coupled interface to the data access layer.
- Encapsulate data access functionality within the data access layer.
- **Decide how to map application entities to data source structures.**
- Consider consolidating data structures.
- Decide how you will manage connections.
- Determine how you will handle data exceptions.
- Consider security risks.
- Reduce round trips.
- Consider performance and scalability objectives.

Specific Design Issues

https://msdn.microsoft.com/en-us/library/ee658127.aspx

- Batching
- Binary Large Objects (BLOBs)
- Connections
- Data Format
- Exception Management
- Object Relational Mapping (http://nhibernate.info/ as an example)
- Queries
- Stored Procedures
- Stored Procedures vs. Dynamic SQL
- Transactions
- Validation
- XML

To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem

https://www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/?from=http%3A%2F%2Fresearch.microsoft.com%2Fapps%2Fpubs%2Fdefault.aspx%3Fid%3D64525

LINKÖPINGS UNIVERSITET

Performance Considerations

https://msdn.microsoft.com/en-us/library/ee658127.aspx

- Use connection pooling and tune performance based on results obtained by running simulated load scenarios.
  - https://msdn.microsoft.com/en-us/library/bb399543(v=vs.110).aspx
- Consider tuning isolation levels for data queries. If you are building an application with high-throughput requirements, special data operations may be performed at lower isolation levels than the rest of the transaction. Combining isolation levels can have a negative impact on data consistency, so you must carefully analyze this option on a case by case basis.
- Consider batching commands to reduce the number of round trips to the database server.
- Consider using optimistic concurrency with nonvolatile data to mitigate the cost of locking data in the database. This avoids the overhead of locking database rows, including the connection that must be kept open during a lock.
- If using a **DataReader**, use ordinal lookups for faster performance.
  - https://msdn.microsoft.com/en-us/library/haa3afyz(v=vs.110).aspx

Security Considerations

https://msdn.microsoft.com/en-us/library/ee658127.aspx

- When storing passwords, use a salted hash instead of an encrypted version of the password.
- Require that callers send identity information to the data layer for auditing purposes.
- Use parameterized SQL queries and typed parameters to mitigate security issues and reduce the chance of SQL injection attacks succeeding. Do not use string concatenation to build dynamic queries from user input data.


LINQ
Microsoft
.NET
LEARN LINQ
language integrated query

What is LINQ?

https://msdn.microsoft.com/en-us/library/bb308959.aspx

General-purpose query facilities added to the .NET Framework apply to all sources of information, not just relational or XML data. This facility is called .NET Language-Integrated Query (LINQ).

Language Integrated Query (**LINQ**, pronounced "link") is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages, although ports exist for PHP (PHPLinq),JavaScript (linq.js), TypeScript (linq.ts), and ActionScript (ActionLinq) - but none of these ports are strictly equivalent to LINQ in C# for example (where it is a part of the language, not an external library, and where it often addresses a wider range of needs).

LINQ Keywords

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/query-keywords

```
from, where, select, group, into, orderby,
join, let, in, on, equals, by, ascending,
descending.
```

LINKÖPINGS UNIVERSITET

IEnumerable/IEnumerator

https://msdn.microsoft.com/en-us/library/system.collections.ienumerable(v=vs.110).aspx

- IEnumerable Exposes an IEnumerator, which supports a simple iteration over a collection.
- IEnumerable is the base interface for all non-generic collections that can be enumerated.
- For the generic version of this interface see System.Collections.Generic.IEnumerable<T>.
- IEnumerable contains a single method, GetEnumerator, which returns an IEnumerator.
- IEnumerator provides the ability to iterate through the collection by exposing:
  - Current property
  - MoveNext method
  - Reset method.
- IEnumerable and IEnumerator enables use of foreach.

LINKÖPINGS UNIVERSITET

LINQ

https://msdn.microsoft.com/en-us/library/ee658127.aspx

```
using System;
using System.Linq;
using System.Collections.Generic;

class app {
  static void Main() {
    string[] names = { "Burke", "Connor", "Frank",
                       "Everett", "Albert", "George",
                       "Harris", "David" };

    IEnumerable<string> query = from s in names
                                where s.Length == 5
                                orderby s
                                select s.ToUpper();

    foreach (string item in query)
      Console.WriteLine(item);
  }
}
```

LINKÖPINGS UNIVERSITET

LINQ - Example 01/Group By

```
int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7,
2, 0 };

var numberGroups = from n in numbers
group n by n % 5 into g select new {
Remainder = g.Key, Numbers = g };

foreach (var g in numberGroups)
{
   Console.WriteLine("Numbers with a
   remainder of {0} when divided by 5:",
   g.Remainder);
   foreach (var n in g.Numbers)
   {
      Console.WriteLine(n);
   }
 }
```

```
Numbers with a remainder of 0 when divided by 5:
5
0
Numbers with a remainder of 4 when divided by 5:
4
9
Numbers with a remainder of 1 when divided by 5:
1
6
Numbers with a remainder of 3 when divided by 5:
3
8
Numbers with a remainder of 2 when divided by 5:
7
2
```

LINKÖPINGS UNIVERSITET

LINQ - Example 02/Group By

```
string[] words = { "blueberry",
"chimpanzee", "abacus", "banana",
"apple", "cheese" };
var wordGroups = from w in words group w
by w[0] into g select new { FirstLetter =
g.Key, Words = g };

foreach (var g in wordGroups)
{
  Console.WriteLine("Words that start
  with the letter '{0}':",g.FirstLetter);
  foreach (var w in g.Words)
  {
      Console.WriteLine(w);
  }
}
```

```
Words that start with the letter 'b':
blueberry
banana
Words that start with the letter 'c':
chimpanzee
cheese
Words that start with the letter 'a':
abacus
apple
```

LINKÖPINGS UNIVERSITET

Linq - example 03/Group By

```
List<Product> products = GetProductList();

var orderGroups = from p in products group p by p.Category into g select new {
Category = g.Key, Products = g };

ObjectDumper.Write(orderGroups);
```

ObjectDumper(A handy tool not part of the .NET core)
https://www.codingame.com/playgrounds/2098/how-to-dump-objects-in-c/using-objectdumper

LINKÖPINGS UNIVERSITET

LINQ - Example 04/GroupBy - Nested

```
List<Customer> customers = GetCustomerList();
var customerOrderGroups = from c in customers select new
                {
                     CompanyName = c.CompanyName,
                     YearGroups = from o in c.Orders group o by o.OrderDate.Year into yg select new
                             {
                                Year = yg.Key,
                               MonthGroups = from o in yg group o by o.OrderDate.Month into mg select new
                                {
                                     Month = mg.Key, Orders = mg
                                 }
                             }
                 };

ObjectDumper.Write(customerOrderGroups);
```

LINKÖPINGS UNIVERSITET

LINQ - Example 05/GroupBy - Comparer

```
public void Linq44()
{
   string[] anagrams = { "from   ", " salt",
   " earn ", "  last   ", " near ", " form "
   };

   var orderGroups = anagrams.GroupBy(w =>
   w.Trim(), new AnagramEqualityComparer());

   ObjectDumper.Write(orderGroups);
}
```

```
public class AnagramEqualityComparer : IEqualityComparer<string>
   {
       public bool Equals(string x, string y)
       {
           return getCanonicalString(x) ==getCanonicalString(y);
       }

       public int GetHashCode(string obj)
       {
           return getCanonicalString(obj).GetHashCode();
       }

       private string getCanonicalString(string word)
       {
           char[] wordChars = word.ToCharArray();
           Array.Sort<char>(wordChars);
           return new string(wordChars);
       }
   }
```

LINKÖPINGS UNIVERSITET

LINQ - Example 06/Element operator

```
List<Product> products = GetProductList();

    Product product12 = (
        from p in products
        where p.ProductID == 12
        select p)
        .First();

ObjectDumper.Write(product12);
```

LINQ - Example 07/Element operator

```
int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

    int fourthLowNum = (
        from n in numbers
        where n > 5
        select n)
        .ElementAt(1);  // second number is index 1 because sequences use 0-based
indexing

    Console.WriteLine("Second number > 5: {0}", fourthLowNum);
```

LINKÖPINGS UNIVERSITET

LINQ - Example 08/Ordering

```
string[] words = { "cherry", "apple", "blueberry" };

    var sortedWords =
        from w in words
        orderby w
        select w;

    Console.WriteLine("The sorted list of words:");
    foreach (var w in sortedWords)
    {
        Console.WriteLine(w);
    }
```

LINKÖPINGS UNIVERSITET

LINQ - Example 09/Ordering

```
string[] words = { "cherry", "apple", "blueberry" };

    var sortedWords =
        from w in words
        orderby w.Length
        select w;

    Console.WriteLine("The sorted list of words (by length):");
    foreach (var w in sortedWords)
    {
        Console.WriteLine(w);
    }
```

LINKÖPINGS UNIVERSITET

LINQ - Example 10/Ordering

```
List<Product> products = GetProductList();

    var sortedProducts =
        from p in products
        orderby p.ProductName
        select p;

    ObjectDumper.Write(sortedProducts);
```

LINQ - Example 11/Ordering

```
public void Linq31()
{
    string[] words = { "aPPLE", "AbAcUs", "bRaNcH", "BlUeBeRrY", "ClOvEr", "cHeRry" };

    var sortedWords = words.OrderBy(a => a, new CaseInsensitiveComparer());

    ObjectDumper.Write(sortedWords);
}

public class CaseInsensitiveComparer : IComparer<string>
{
    public int Compare(string x, string y)
    {
        return string.Compare(x, y, StringComparison.OrdinalIgnoreCase);
    }
}
```

https://docs.microsoft.com/en-us/dotnet/api/system.stringcomparer?view=netframework-4.7.2

LINQ - Example 12/Ordering

```
double[] doubles = { 1.7, 2.3, 1.9, 4.1, 2.9 };

var sortedDoubles = from d in doubles orderby d descending select d;

Console.WriteLine("The doubles from highest to lowest:");

foreach (var d in sortedDoubles)
{
    Console.WriteLine(d);
}
```

LINQ - Example 13/Ordering

```
string[] digits = { "zero", "one", "two", "three", "four", "five", "six", "seven",
"eight", "nine" };

var sortedDigits = from d in digits orderby d.Length, d select d;

Console.WriteLine("Sorted digits:");

foreach (var d in sortedDigits)
{
    Console.WriteLine(d);
}
```

LINKÖPINGS UNIVERSITET

LINQ - Example 14/Aggregate

```
var numbers = new List<int> { 6, 2, 8, 3 };
int sum = numbers.Aggregate((result, item) => result + item);
// sum: (((6+2)+8)+3) = 19
```

LINKÖPINGS UNIVERSITET

Linq - example 15/Aggregate

```
double[] doubles = { 1.7, 2.3, 1.9, 4.1, 2.9 };

double product = doubles.Aggregate((runningProduct, nextFactor) => runningProduct *
nextFactor);

Console.WriteLine("Total product of all numbers: {0}", product);
```

LINKÖPINGS UNIVERSITET

LINQ - Example 16/Cross Join

```
string[] categories = new string[]{
        "Beverages",
        "Condiments",
        "Vegetables",
        "Dairy Products",
        "Seafood" };

List<Product> products = GetProductList();

var q = from c in categories join p in products on c equals p.Category select new {
Category = c, ProductName = p.ProductName };

foreach (var v in q)
{
        Console.WriteLine(v.ProductName + ": " + v.Category);
}
```

LINKÖPINGS UNIVERSITET

LINQ - Example 17/Grouped Join

```
string[] categories = new string[]{
        "Beverages",
        "Condiments",
        "Vegetables",
        "Dairy Products",
        "Seafood" };

    List<Product> products = GetProductList();
    var q =
        from c in categories
        join p in products on c equals p.Category into ps
        select new { Category = c, Products = ps };
    foreach (var v in q)
    {
        Console.WriteLine(v.Category + ":");
        foreach (var p in v.Products)
        {
            Console.WriteLine("   " + p.ProductName);
        }
    }
```

LINQ - Inner Join

https://docs.microsoft.com/en-us/dotnet/csharp/linq/perform-inner-joins

LINQ - Left Outer Join

https://docs.microsoft.com/en-us/dotnet/csharp/linq/perform-left-outer-joins

LINQ - Grouped Joins

https://docs.microsoft.com/en-us/dotnet/csharp/linq/perform-grouped-joins

More examples @

https://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b

XML write

https://msdn.microsoft.com/en-us/library/system.xml.linq.xdocument(v=vs.110).aspx

```
XDocument doc = new XDocument(
    new XElement("Root",
        new XElement("Child", "content")
    )
);
doc.Save("Root.xml");
Console.WriteLine(File.ReadAllText("Root.xml"));
```

```
<!-- Result -->
<?xml version="1.0" encoding="utf-8"?>
<Root>
  <Child>content</Child>
</Root>
```

LINKÖPINGS UNIVERSITET

XML read

https://msdn.microsoft.com/en-us/library/system.xml.linq.xdocument(v=vs.110).aspx

```
XDocument document = XDocument.Load("D:\\Data.xml");
var books = from r in document.Descendants("book")
select new
    {
            Author = r.Element("author").Value,
            Title = r.Element("title").Value,
            Genere = r.Element("genre").Value,
            Price = r.Element("price").Value,
            PublishDate = r.Element("publish_date").Value,
            Description = r.Element("description").Value,

   };

foreach (var r in books)
{
    Console.WriteLine(r.PublishDate + r.Title + r.Author );
}
```

LINKÖPINGS UNIVERSITET

JSON write

https://www.newtonsoft.com/json/help/html/WriteToJsonFile.htm

```
JObject videogameRatings = new JObject(
    new JProperty("Halo", 9),
    new JProperty("Starcraft", 9),
    new JProperty("Call of Duty", 7.5));

File.WriteAllText(@"c:\videogames.json", videogameRatings.ToString());

// write JSON directly to a file
using (StreamWriter file = File.CreateText(@"c:\videogames.json"))
using (JsonTextWriter writer = new JsonTextWriter(file))
{
    videogameRatings.WriteTo(writer);
}
```

https://www.newtonsoft.com/json/help/html/LINQtoJSON.htm
https://www.newtonsoft.com/json/help/html/Introduction.htm

LINKÖPINGS UNIVERSITET

JObject

https://www.newtonsoft.com/json/help/html/QueryingLINQtoJSON.htm

```
var postTitles =
    from p in rss["channel"]["item"]
    select (string)p["title"];

foreach (var item in postTitles)
{
    Console.WriteLine(item);
}

var categories =
    from c in rss["channel"]["item"].SelectMany(i => i["categories"]).Values<string>()
    group c by c
    into g
    orderby g.Count() descending
    select new { Category = g.Key, Count = g.Count() };

foreach (var c in categories)
{
    Console.WriteLine(c.Category + " - Count: " + c.Count);
}
```

LINKÖPINGS UNIVERSITET

Executing SQL commands in C# code

https://msdn.microsoft.com/en-us/library/fksx3b4f.aspx

```
SqlConnection sqlConnection1 = new SqlConnection("Your Connection String");
SqlCommand cmd = new SqlCommand();
SqlDataReader reader;

cmd.CommandText = "SELECT * FROM Customers";
cmd.CommandType = CommandType.Text;
cmd.Connection = sqlConnection1;

sqlConnection1.Open();

reader = cmd.ExecuteReader();
// Data is accessible through the DataReader object here.
//In order to use linq to access the data, it first needs to be stored as a list.

sqlConnection1.Close();
```

LINKÖPINGS UNIVERSITET

SQLDataReader

https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader(v=vs.110).aspx

- Provides a way of reading a forward-only stream of rows from a SQL Server database.
- This class cannot be inherited.
- To create a SqlDataReader, you must call the ExecuteReader method of the SqlCommand object.
- While the SqlDataReader is being used, no other operation can be performed until the close method is called.
- Changes made to a result set by another process or thread while data is being read may be visible to the user of the SqlDataReader.

Using ORM with LINQ(LinqConnect)

https://www.devart.com/dotconnect/mysql/articles/tutorial_linq.html

LinqConnect is an ORM which has been developed for LINQ to be used with MySQL.

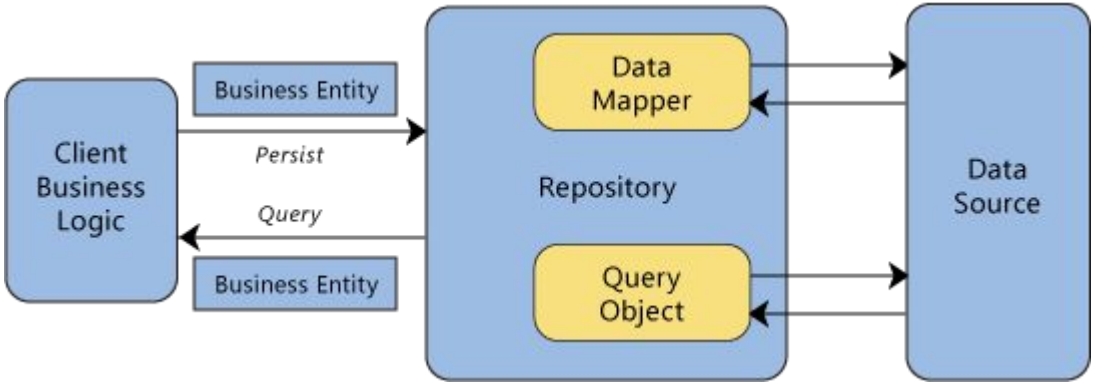
```
CrmDataContext context = new CrmDataContext();
var query = from it in context.Companies
            orderby it.CompanyID
            select it;

foreach (Company comp in query)
    Console.WriteLine("{0} | {1} | {2}", comp.CompanyID, comp.CompanyName, comp.Country);

Console.ReadLine();
```

LINKÖPINGS UNIVERSITET

Repository Pattern

https://msdn.microsoft.com/en-us/library/ff649690.aspx

The repository mediates between the data source layer and the business layers of the application. It queries the data source for the data, maps the data from the data source to a business entity, and persists changes in the business entity to the data source. A repository separates the business logic from the interactions with the underlying data source or Web service.



LINKÖPINGS UNIVERSITET

Thanks for listening!

LINKÖPINGS UNIVERSITET