Mahindra Satyam

Workbook for LINQ

A beginner's guide to effective programming

(In C#.Net)

Why This Module

After two decades, the industry has reached a stable point in the evolution of object oriented programming technologies. Programmers now take for granted - features like classes, objects, and methods. In looking at the current and next generation of technologies, it has become apparent that the next big challenge in programming technology is to reduce the complexity of accessing and integrating information that is not natively defined using OO technology. The two most common sources of non-OO information are relational databases and XML.

Rather than add relational or XML-specific features to our programming languages and runtime, with the LINQ project we have taken a more general approach and are adding general purpose query facilities to the .NET Framework that apply to all sources of information, not just relational or XML data. This facility is called .NET Language Integrated Query (LINQ).

We use the term *language integrated query* to indicate that query is an integrated feature of the developer's primary programming languages (e.g., C#, Visual Basic). Language integrated query allows *query expressions* to benefit from the rich metadata, compile-time syntax checking, static typing and IntelliSense that was previously available only to imperative code. Language integrated query also allows a single general purpose declarative query facility to be applied to all in-memory information, not just information from external sources.

Guide to Use this Workbook

Conventions Used

Convention	Description
Topic	Indicates the Topic Statement being discussed.
Estimated Time	Gives an idea of estimated time needed to understand the Topic and complete the Practice session.
Presentation	Gives a brief introduction about the Topic.
Scenario	Gives a real time situation in which the Topic is used.
Demonstration/Code Snippet	Gives an implementation of the Topic along with Screenshots and real time code.
Code in Italic	Represents a few lines of code in Italics which is generated by the System related to that particular event.
// OR '	Represents a few lines of code (Code Snippet) from the complete program which describes the Topic.
Context	Explains when this Topic can be used in a particular Application.
Practice Session	Gives a practice example for the participant to implement the Topic, which gives him a brief idea how to develop an Application using the Topic.
Check list	Lists the brief contents of the Topic.
Common Errors	Lists the common errors that occur while developing the Application.

Exceptions	Lists the exceptions which result from the execution of the Application.
Lessons Learnt	Lists the lessons learnt from the article of the workbook.
Best Practices	Lists the best ways for the efficient development of the Application.
Notes	Gives important information related to the Topic in form of a note

Mahindra Satyam

Contents

1.0 Introduction to LINQ

Topics

Topic: LINQ Basics

Estimated Time: 20 min.

Objectives: This module will familiarize the participant with

- LINQ
- LINQ Architecture

Presentation:

- LINQ stands for **Language Integrated Query**
- Provides uniform programming model for any type of data
- Provides ability to query and manipulate data with a consistent model independent of the data source.
- Provides general purpose query facilities to .Net Platform.
- Strongly Integrated with Dot Net Languages like C# and VB.

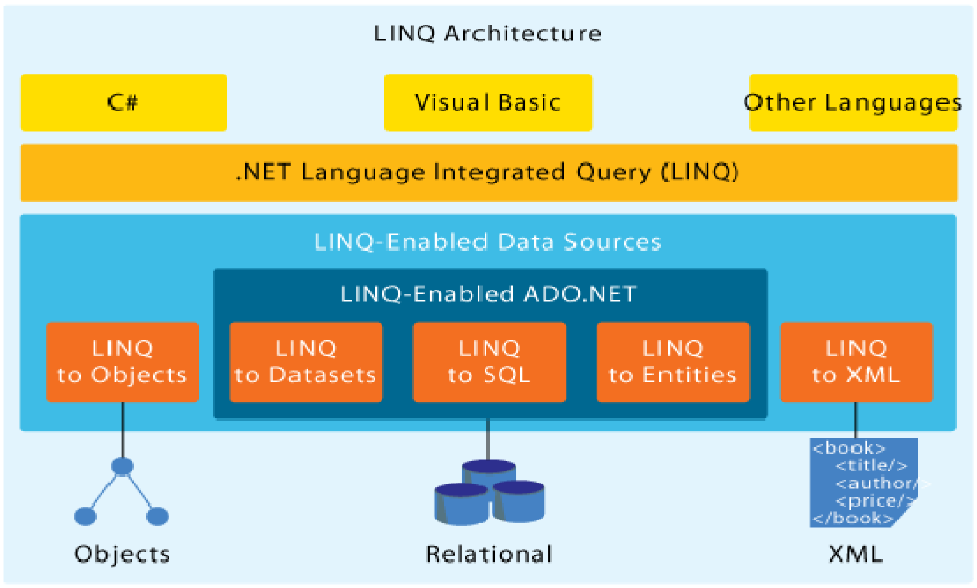


Figure 1.1-1: LINQ- Architecture

- LINQ can be used to query and transform the data in

- ➢ SQL Databases
- ➢ XML Document
- ➢ ADO.Net Datasets
- ➢ .Net Collections

- LINQ Query Operation contains three parts
  - o Obtain the data source
  - o Create the query
  - o Execute the Query

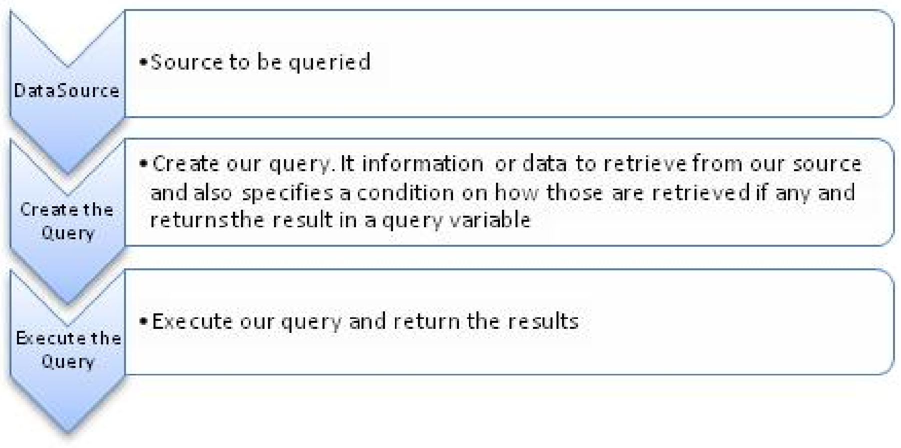

DataSource
•Source to be queried
Create the Query
•Create our query. It information or data to retrieve from our source and also specifies a condition on how those are retrieved if any and returnsthe result in a query variable
Execute the Query
•Execute our query and return the results

Figure 1.1-1: LINQ Query Operation.

Scenario:

Mr. David a publisher, preparing a journal for "Satyam Computer Services Ltd", intends to publish the various designations of the associates in the organization.

Context:

- LINQ can be used for retrieval of data from any type of data source

Practice Session:

- Mr. David owns a Software Organization. He wants to find out who the highest paid employee in his company is. Implement this using LINQ query pattern.
- Suggest real-life scenarios where LINQ is likely to be employed.

Check list:

- Advantages of LINQ.

Common Errors:

- Not including System.Linq namespace

Exceptions:

- .Net cannot identify query patten and its implementation in absence of System.Linq namespace.

Lessons Learnt:

☑ Data doesn’t get retrieved writing a query
☑ A query has to be executed to retrieve data from the data source.

Notes:

- Query expression contains three clauses **from, where and select.**
- The **from** clause specifies the data source, **where** clause applies the filter and **select** clause specifies the types of returned elements.
- The actual execution of the query starts when iteration is done over the query, variable in for each statement.
- The query doesn’t get executed until the program calls the data from the query object. Thus LINQ uses deferred execution.

Topic: LINQ-Query Expression

Estimated Time: 30 min.

Objectives: At the end of the activity, the participant will be able to understand:

- Query Expression Basics.
- Execution of Query Expression.

Presentation:

- Query expression is a query expressed in query syntax
- Can be used to query and to transform data from any LINQ-enabled data source (IEnumerable<T>).
- Basic Form of Query Expression is

```
from [identifier] in [source collection]
let [expression]
where [boolean expression]
order by [[expression](ascending/descending)], [optionally repeat]
select [expression]
group [expression] by [expression] into [expression]
```

- At compile time, the query expression is converted into standard operator method calls, as per C# specifications.
- The result of a query expression is a query object, which represents the commands needed to execute the query(an Expression Tree)
- Execution of Query expression can be done in two different modes
  a. Deferred execution
  b. Immediate execution.
- The query does not execute until the program requests data from the query object. This behavior is known as **deferred execution** and is a powerful feature of LINQ. Deferred execution facilitates the composability of queries and allows applications to pass queries around as data.
- A query object implements the IQueryable<T> interface, which inherits from IEnumerable<T> - meaning we can still treat the query object like a collection. The query executes when we iterate the collection.

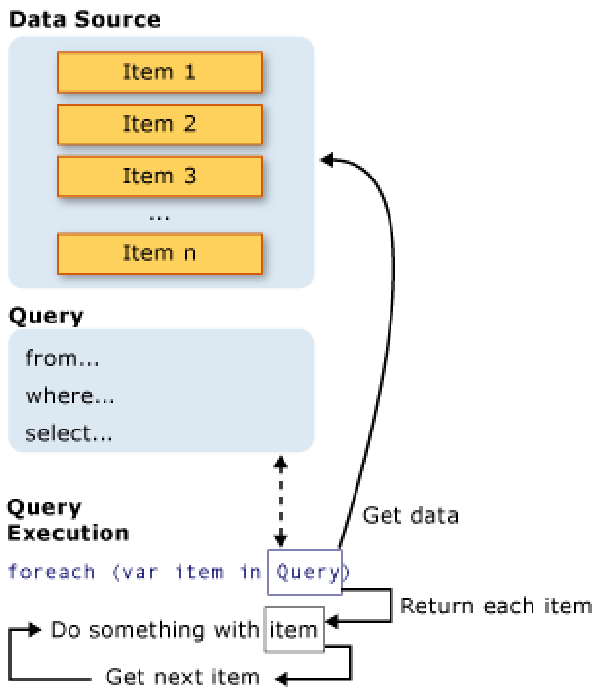
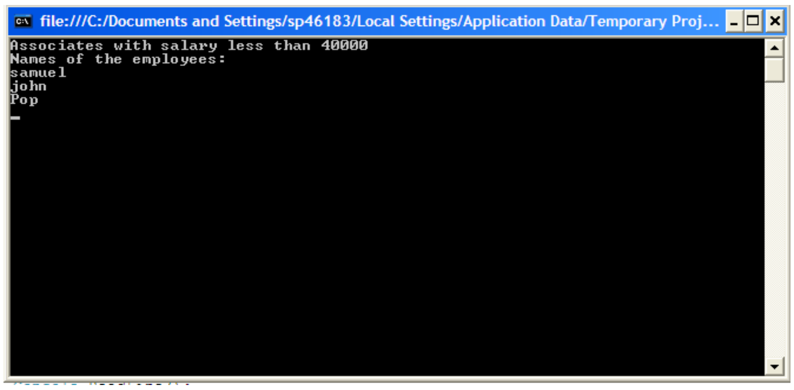


Figure 1.2-1: Query Execution

Scenario:

Mr. David owns a software company. He wants to view the details of the employees whose salary is below 40,000. Build a console application which displays the employees details whose salary is less than 40,000.

Demonstration/Code Snippet:


file:///C:/Documents and Settings/sp46183/Local Settings/Application Data/Temporary Proj...

```
Associates with salary less than 40000
Names of the employees:
samuel
john
Pop
_
```

Figure 1.2-2: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new Console application. Insert the following Code. It includes the corresponding namespaces.

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

Step 2: Define a class " Employee" with ID,Name,Salary.

```
public class employee
        {


            public int ID { get; set; }
            public string Name { get; set; }
            public int Salary { get; set; }


        }
```

Step 3 : Initialise the employee list details with values

```
List<employee> employees = new List<employee>(){

                new employee{
ID=121,Name="samuel",Salary=10000 },
                new
employee{ID=122,Name="john",Salary=20000},
                new
employee{ID=123,Name="Pop",Salary=30000},
                new
employee{ID=124,Name="Stephen",Salary=40000},
                new
employee{ID=125,Name="Abraham",Salary=50000}

           };
```

Step 4 : Querying and displaying the employees with salary less than 40000

```
var Empdetails = from emp in employees
                              where emp.Salary < 40000
                              select emp;
            Console.WriteLine("Associates with salary
less than 40000");
            Console.WriteLine("Names of the
employees:");
           foreach (employee emp in Empdetails)
           {


               Console.WriteLine(emp.Name);

           }
           Console.ReadLine();
```

Step 5 : Include the Complete Program in Program.cs

```
namespace ConsoleApplication1
{
    class Program
    {
       public class employee
        {
             public int ID { get; set; }
             public string Name { get; set; }
             public int Salary { get; set; }
        }
   static void Main(string[] args)
     {

   //Initialisation
  new employee{ID=121,Name="samuel",Salary=10000};
           List<employee> employees = new
List<employee>(){
new employee{ ID=121,Name="samuel",Salary=10000 },
new employee{ID=122,Name="john",Salary=20000},
new employee{ID=123,Name="Pop",Salary=30000},
new employee{ID=124,Name="Stephen",Salary=40000},
new employee{ID=125,Name="Abraham",Salary=50000}
               };
 //Querying
var Empdetails = from emp in employees
                 where emp.Salary < 40000
                 select emp;
Console.WriteLine("Associates with salary less than
40000");
 Console.WriteLine("Names of the employees:");
 foreach (employee emp in Empdetails)
           {

               Console.WriteLine(emp.Name);

           }
           Console.ReadLine();}}}
```

Context:

- To read data and manipulate data from any LINQ-enabled data source

Practice Session:

- Mr. David, a publisher preparing a journal for “Satyam Computer Services Ltd”, intends to publish the various designations of the associates in the organization.
- Display the details using the query expression.

Check list:

- List of operators which can be deferred.
- How to achieve immediate execution.

Common Errors:

- Incorrect or insufficient namespaces being imported into the application.
- Syntax errors while writing the query expressions.

Exceptions:

- An exception might occur if we do our coding with respect to a data source which is different from the imported namespace.

Lessons Learnt:

☑ The benefit of deferred execution is that it allows the same LINQ query to be executed multiple times, giving you the latest results

☑ Sometimes this may lead to performance problems as the query gets executed each time.

Best Practices:

- Use the Query Expression syntax wherever possible, it’s easier to read.
- If you need to mix the Extension Methods with Query Expressions, put them at the end.
- Keep each part of the Query Expression on separate lines to allow you to individually comment out an individual clause for debugging.

Topic: LINQ-Standard Operators

Estimated Time: 30 min.

Objectives: At the end of the activity, the participant will be able to understand:

- Standard operator basics.
- Classification of standard Operators.

Presentation:

- A standard Query operator is an API that enables to query any collection.
- It consists of methods declared in System. Query Sequence static class.
- They operate on sequences (IEnumerable<T>).
- Classification of Query Operators.

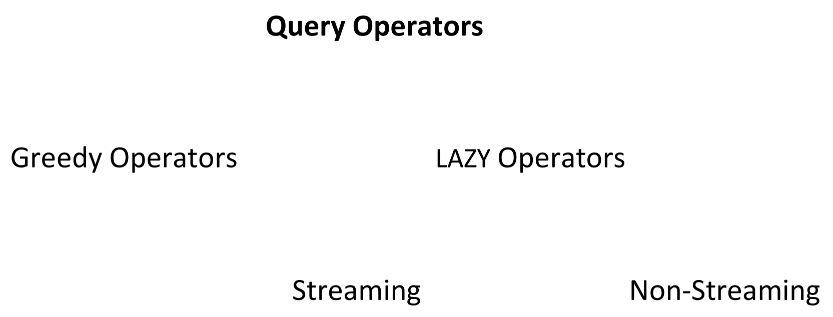


Figure 1.3-1: Standard Operators Classification

- Lazy operators follow deferred execution and greedy operators follow immediate execution.
- Lazy operators are further classified in to streaming (group by) and non- streaming operators (e.g.: order by).
- A non-streaming operator will need to evaluate all the data in a sequence before it can produce a result.
- Different types of operators depending on their functionality are:

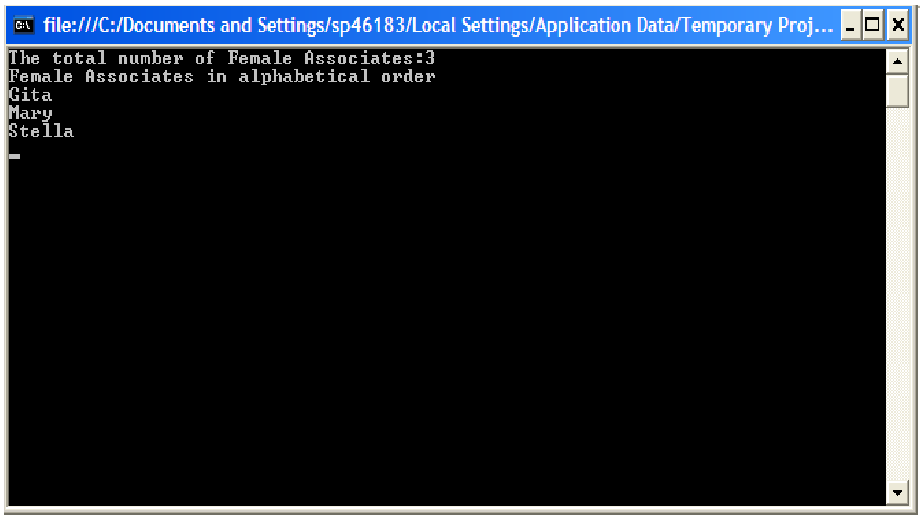
❖ **Filtering**	(where, oftype)
❖ **Projecting**	(select,SelectMany)
❖ **Joining**	(equals)
❖ **Ordering**	(orderby,Thenby)
❖ **Grouping**	(groupby,ToLookUp)
❖ **Conversions**	(ToList)

❖ **Sets**	(distinct,union)
❖ **Aggregation**	(count,Max)
❖ **Quantifiers**	(all,any)
❖ **Generation**	(eg:Empty)
❖ **Elements**	(eg: first, last)

Scenario:

Mr. David owns SmartTech Hardware Solutions Company. He wants to view the details of all the Female associates of his company. Build a Console application which displays the total number of all the female associates along with their details in alphabetical order of their names. The employee details include Employee Id , Empoyee name, Gender and Salary. Use Standard Query operators for displaying the details.

Demonstration/Code Snippet:


file:///C:/Documents and Settings/sp46183/Local Settings/Application Data/Temporary Proj...

```
The total number of Female Associates:3
Female Associates in alphabetical order
Gita
Mary
Stella
```

Figure 1.2-2: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new Console application. Insert the following Code. It includes the corresponding namespaces.

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

Step 2: Create a class " Employee" with ID,Name,Salary.

```
public class employee
        {

            public int ID { get; set; }
            public string Name { get; set; }
            public int Salary { get; set; }
            public char Gender { get; set; }

        }
```

Step 3 : Initialise the employee details with values

```
List<employee> employees = new List<employee>(){
new employee{ID=121,Name="samuel",Salary=10000,Gender=
'M' }
new employee{ID=122,Name="Mary",Salary=20000, Gender=
'F'},
new employee{ID=123,Name="Stella",Salary=30000,Gender=
'F'},
new employee{ID=124,Name="Gita",Salary=40000,Gender=
'F'},
newemployee{ID=125,Name="Abraham",Salary=50000,Gender=
'M'}
   };
```

Step 4 : Querying and displaying the employees with salary less than 40000

```
var Empdetails = from emp in employees
                            where emp.Gender == 'F'
                            orderby emp.Name
                            select emp.Name;
           var count = Empdetails.Count();
           Console.WriteLine("The total number of
Female Associates:{0}",count);
          Console.WriteLine("Female Associates in
alphabetical order");
          foreach (string nam in Empdetails)
           {

               Console.WriteLine(nam);

           }
           Console.ReadLine();
```

Step 5 : Include the Complete Program in Program.cs

```
namespace ConsoleApplication1
{
    class Program
    {
        public class employee
        {
            public int ID { get; set; }
            public string Name { get; set; }
            public int Salary { get; set; }
            public char Gender { get; set; }
        }
        static void Main(string[] args)
        {
           //Initialisation
 List<employee> employees = new List<employee>(){
 new employee{ ID=121,Name="samuel",Salary=10000,Gender=
'M' },
 new employee{ID=122,Name="Mary",Salary=20000, Gender=
'F'},
 new employee{ID=123,Name="Stella",Salary=30000,Gender=
'F'},
new employee{ID=124,Name="Gita",Salary=40000,Gender=
'F'},
new employee{ID=125,Name="Abraham",Salary=50000,Gender=
'M'}
 };       //Querying
           var Empdetails = from emp in employees
                           where emp.Gender == 'F'
                            orderby emp.Name
                            select emp.Name;
           var count = Empdetails.Count()
Console.WriteLine("The total number of Female
Associates:{0}",count);
Console.WriteLine("Female Associates in alphabetical
order");
 foreach (string nam in Empdetails)
           {
               Console.WriteLine(nam);
           }
           Console.ReadLine();

        }
    }
}
```

Context:

- To use standard query operators for output of data from any data source.

Practice Session:

- A book seller wants to sort out the books in alphabetical order and arrange them in separate shelves according to the size of the books. Write a program to sort the books using standard query operators

Check list:

- Other standard operators
- Difference between select and Select Many

.

Common Errors:

- Incorrect or insufficient namespaces being imported into the application.
- Syntax errors while writing the query expressions.

Exceptions:

- An exception might occur if we do our coding with respect to a data source which is different from the imported namespace.
- Trying to use invalid Standard Operator.

Lessons Learnt:

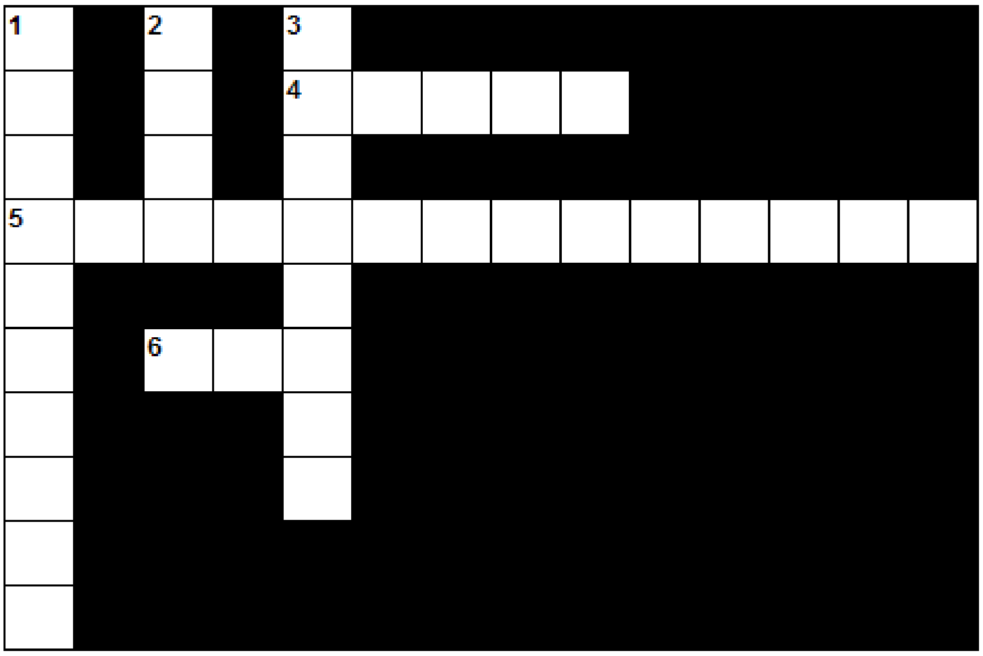
☑ Querying the data using different set of LINQ query operators.

Best Practices:

- Use the Query
- Expression syntax instead of method based wherever possible, it's easier to read.

Crossword: Unit-1

Estimated Time: 10 min.



Across:

5. The result of an query expression(14)

6. Implicitly typed variable(3)

4. A Generational operator(5)

Down:

1. A query object implements the interface (10).

2. One of the primary partitioning operator (4)

2. Linq Implements _______________ execution.(8)

.

2.0 LINQ to Objects

Topics

Topic: LINQ-Objects

Estimated Time: 30 min.

Objectives: At the end of the activity, the participant will be able to understand:

- Introduction to LINQ to Objects.
- Working with LINQ to objects.

Presentation:

- LINQ to Objects refers to the use of LINQ queries with any IEnumerable or IEnumerable collection directly, without the use of an intermediate LINQ provider or API such as LINQ to SQL or LINQ to XML.
- It's a new approach for collections. The collection may be user defined or returned by a .Net Framework API.
- It's an alternative way for the traditional for each loop way that specified how to retrieve data from collection.
- LINQ can be used to query and transform strings and collections of strings. LINQ queries can be combined with traditional string functions and regular expressions.
- It can be especially useful with semi-structured data in text files.

Scenario:

- Mr. Samuel is the Chief Doctor in Apollo Health clinic. He wants to view the details of the patients which include in & out details, Cause of admit etc. He also wants to view the following details in particular.

  - ❖ The details of the patient "scott"
  - ❖ The details of all those patients who are admitted under regular chekup before 2nd FEB,2009
  - ❖ Details of all the patients excluding first "n" number of patients
  - ❖ Details of all the patients who are admitted before 1st FEB 2009.

Demonstration/Code Snippet:

Step 1: Open Visual Studio and create a new Console application. Insert the following Code in program.cs. It includes the corresponding namespaces.

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

Step 2: Create a class patientrecords in PatientRecords.cs

```
class PatientRecords
    {
        public int PRecordID { get; set; }
        public string PId { get; set; }
        public string PName { get; set; }
        public string PDisease { get; set; }
        public DateTime PInDate { get; set; }
        public DateTime POutDate { get; set; }
        public String PDetails()
        {
            return String.Format("Record
Number:{0},PId:{1},Name:{2},Cause:{3},InDate:{4},OutDate:{5}"
,PRecordID,PId,PName,PDisease,PInDate.Date ,POutDate.Date  );
        }
    }
```

Step 3 : Inserting the patients details under program.cs

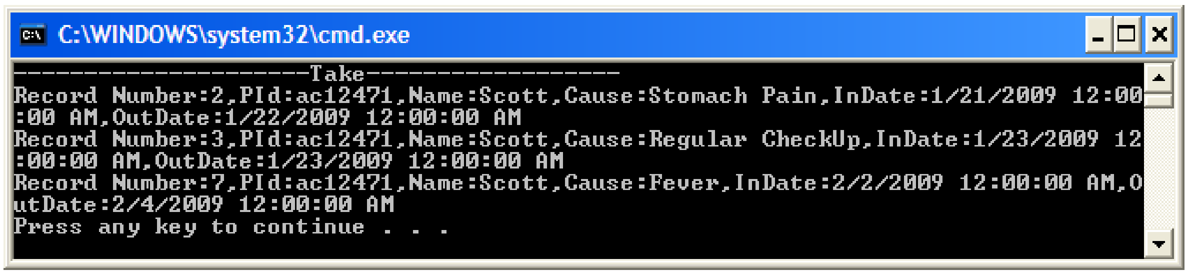
```
List<PatientRecords> PRecords = new List<PatientRecords>
            {
                new PatientRecords { PRecordID=1, PId="ab12451",
PName="Don", PDisease="Stomach Pain", PInDate=new
DateTime(2009,1,21),POutDate=new DateTime(2009,1,23)},
                new PatientRecords { PRecordID=2, PId="ac12471",
PName="Scott", PDisease="Stomach Pain", PInDate=new
DateTime(2009,1,21),POutDate=new DateTime(2009,1,22)},
                new PatientRecords { PRecordID=3, PId="ac12471",
PName="Scott", PDisease="Regular CheckUp", PInDate=new
DateTime(2009,1,23), POutDate=new DateTime(2009,1,23)},
                new PatientRecords { PRecordID=4, PId="ar12534",
PName="Saxena", PDisease="Fever", PInDate=new
DateTime(2009,1,26),POutDate=new DateTime(2009,1,27)},
                new PatientRecords { PRecordID=5, PId="ab12451",
PName="Don", PDisease="Regular CheckUp", PInDate=new DateTime(2009,1,26),
POutDate=new DateTime(2009,1,26)},
                new PatientRecords { PRecordID=6, PId="ae14521",
PName="Claire", PDisease="Tooth Decay", PInDate=new DateTime(2009,1,27),
POutDate=new DateTime(2009,1,29)},
                new PatientRecords { PRecordID=7, PId="ac12471",
PName="Scott", PDisease="Fever", PInDate=new DateTime(2009,2,2),
POutDate=new DateTime(2009,2,4)},
                new PatientRecords { PRecordID=8, PId="ar12534",
PName="Saxena", PDisease="Regular CheckUp", PInDate=new
DateTime(2009,2,3), POutDate=new DateTime(2009,2,3)},
                new PatientRecords { PRecordID=9, PId="av12451",
PName="Marie", PDisease="Fever", PInDate=new DateTime(2009,2,5),
POutDate=new DateTime(2009,2,6)},
                new PatientRecords { PRecordID=10,PId="ag14512",
PName="Kalorine", PDisease="Surgery", PInDate=new DateTime(2009,2,5),
POutDate=new DateTime(2009,2,9)},
                new PatientRecords { PRecordID=11,PId="ay15442",
PName="Kate", PDisease="Regular CheckUp", PInDate=new DateTime(2009,2,7),
POutDate=new DateTime(2009,2,7)},
                new PatientRecords { PRecordID=12,PId="av12451",
```

Step 4 : Querying for all the cases using LINQ and displaying the details.

```
//Select the First 3 records in PatientRecords for
the Patient Name with scott
            Console.WriteLine("---------------------Take--------------
----");
            var pick1 = (from records in PRecords
                         where records.PName == "Scott"
                         select records).Take(3);

            foreach (var item in pick1)
                Console.WriteLine(item.PDetails(), "\n");
```

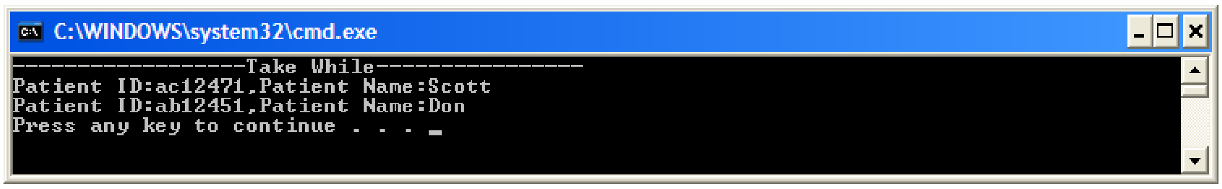
Output for the above:


C:\WINDOWS\system32\cmd.exe

```
----------------------Take------------------
Record Number:2,PId:ac12471,Name:Scott,Cause:Stomach Pain,InDate:1/21/2009 12:00
:00 AM,OutDate:1/22/2009 12:00:00 AM
Record Number:3,PId:ac12471,Name:Scott,Cause:Regular CheckUp,InDate:1/23/2009 12
:00:00 AM,OutDate:1/23/2009 12:00:00 AM
Record Number:7,PId:ac12471,Name:Scott,Cause:Fever,InDate:2/2/2009 12:00:00 AM,O
utDate:2/4/2009 12:00:00 AM
Press any key to continue . . .
```

```
//Select the Records with PDisease as Regular CheckUp
before 2nd Feb, 2009
            Console.WriteLine("------------------Take While----------------
");
            var pick2 = (from records in PRecords
                         where records.PDisease == "Regular CheckUp"
                         orderby records.POutDate ascending
                         select records).TakeWhile(n => n.POutDate <
Convert.ToDateTime("2009/2/2"));
```

Output for the above:

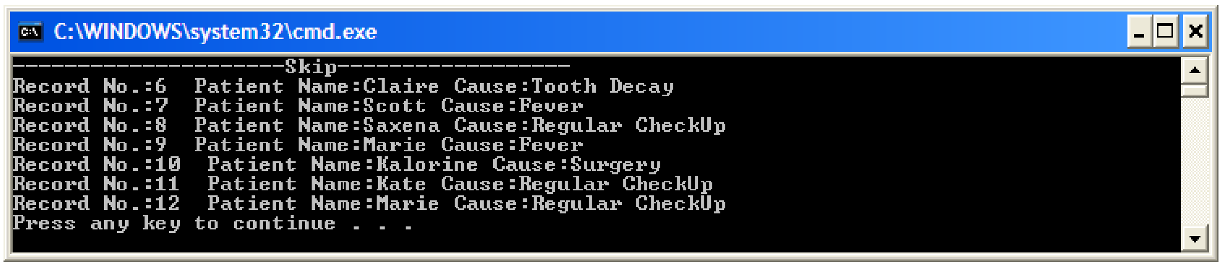

C:\WINDOWS\system32\cmd.exe

```
------------------Take While----------------
Patient ID:ac12471,Patient Name:Scott
Patient ID:ab12451,Patient Name:Don
Press any key to continue . . . _
```

```
//Select Records from PatientRecords except the
First 5 when they are in ascending order(RecordID)
            Console.WriteLine("--------------------Skip---------------
---");
            var pick3 = (from records in PRecords
                         orderby records.PRecordID ascending
                         select records).Skip(5);

            foreach (var item in pick3)
                Console.WriteLine("Record No.:{0}  Patient Name:{1}
Cause:{2}", item.PRecordID, item.PName, item.PDisease);
```

Output for the above:



```
--------------------Skip-----------------
Record No.:6  Patient Name:Claire Cause:Tooth Decay
Record No.:7  Patient Name:Scott Cause:Fever
Record No.:8  Patient Name:Saxena Cause:Regular CheckUp
Record No.:9  Patient Name:Marie Cause:Fever
Record No.:10  Patient Name:Kalorine Cause:Surgery
Record No.:11  Patient Name:Kate Cause:Regular CheckUp
Record No.:12  Patient Name:Marie Cause:Regular CheckUp
Press any key to continue . . .
```

```
//Select Records from PatientRecords except those
whose InDate are after 1st Feb, 2009
            Console.WriteLine("------------------skip While----------------
");
            var pick4 = (from records in PRecords
                         orderby records.PInDate descending
                         select records).SkipWhile(n => n.PInDate >
Convert.ToDateTime("2009/2/1"));

            foreach (var item in pick4)
                Console.WriteLine("Record No.:{0}  Patient Name:{1}
InDate:{2}", item.PRecordID, item.PName, item.PInDate.Date);
```

Context:

- Retrieving the data from a list or collection of objects using LINQ.

Practice Session:

- Mr. Samuel is the editor of Daily Express, a News Edition. As part of review of an article, he wants to find out the number of words in the whole article.

  Hint: Try creating it using lists and split method.

Check list:

- Different operators that can be used for querying the objects.
- Difference between select and select many.

Common Errors:

- Syntax errors.
- Not including the namespace of LINQ
- Mis-spelt field names.

Exceptions:

- System. Invalid Operation Exception: Sequence contains no matching element

Lessons Learnt:

☑ It is suggestible to use LINQ to handle the data from large volume of text

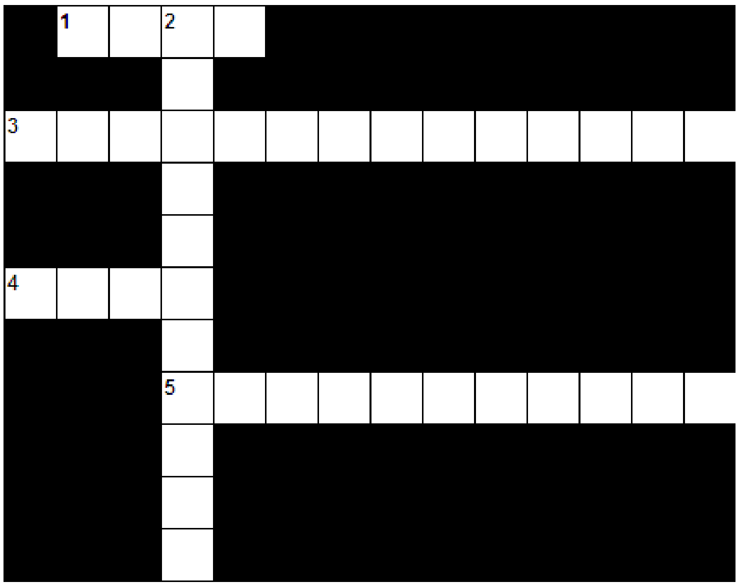
☑ LINQ to objects provide different operators for different scenarios of handling the text files.

Notes:

- The advantages of using LINQ would be the following
- They are more concise and readable, especially when filtering multiple conditions.
- They provide powerful filtering, ordering, and grouping capabilities with a minimum of application code.
- They can be ported to other data sources with little or no modification.

Crossword: Unit-3

Estimated Time: 10 min.



ACROSS:

1. _______________operator should be used to exclude the beginning n number of element s of the list/collection.(4)
3. *All*, *Any*, and *Contains* are _____________ kind of operators.(14)
4. _______________ Operator should be used to pick the beginning n number of element s of the list/collection.(4)
5. ______________ is a method that allow the query to be converted to an instance of IQueryable(11)

DOWN:

2. _______________ is the base interface for all nongeneric enumerators.(11)

3.0 LINQ TO SQL

Topics

Topic: LINQ-SQL Object Model

Estimated Time: 20 min.

Objectives: At the end of the activity participant should understand

- Introduction to LINQ-SQL
- Object model followed in LINQ to SQL.

Presentation:

- LINQ-SQL is an Object – Relational Mapper
- Also known as DLINQ.
- Elements in database are mapped to Object Model.
- Object Model can be created using either O/R designer or SQL Metal command line tool.
- Type Mapping is required in order to translate the data between object model and database.
- Mapping can be done either by applying the attributes or by using the external mapping file.
- Allows querying the SQL database without writing the data access code.
- Translates the Language integrated queries into SQL for execution by the database and then translates the tabular results back into objects.

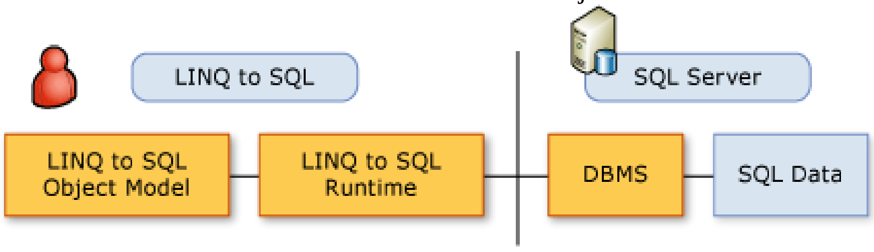


Figure 2.1-1: LINQ Query Operation.

- Supports change tracking for insert, update, and delete operations.
- System.Data.Linq contains classes that support LINQ to SQL applications.
- Maps .NET classes to relational SQL data.

LINQ TO SQL OBJECT MODEL	RELATIONAL DATA MODEL
Entity Class	tables
Class Member	column
Association	Foreign key relationship
Method	Stored procedure/Function

Figure 2.1-2: LINQ relational-object mapping.

Scenario:

- LINQ to SQL can be used to query relational data stores without leaving the syntax or compile-time environment of the local programming language.

Context:

- To retrieve and manipulate data from SQL data base using LINQ

Practice Session:

- Suggest real-life scenarios where LINQ to SQL is likely to be employed.

Check list:

- Limitations of LINQ

Common Errors:

- Incorrect or insufficient namespaces being imported into the application.
- Syntax Errors while using LINQ Query operators.

Exceptions:

- Dot Net accepts <IEnumerable T> type. Hence the relational model data from the database is to be converted into proper type before querying on it.

Lessons Learnt:

☑ LINQ supports all the Key capabilities of a SQL developer.
☑ Using LINQ to SQL, We can query for information, insert, update and delete from tables

Best Practices:

- LINQ to SQL is more efficient in a scenario where in there are less updations made to the database
.

Topic: LINQ-SQLO/R designer

Estimated Time: 30 min.

Objectives:

At the end of the activity, the participant will be able to understand:

- Creation of entity classes using the O/R designer in Visual Studio

Presentation:

- LINQ to Sql classes that are mapped to database tables and views are called entity classes.
- O/R designer provides a visual surface design for creating entity classes and associations based on objects in a database.
- Generates the .dbml file that provides the mapping between the LINQ to SQL classes and database
- Entity classes are created and stored in Linq to Sql classes files(.dbml files)
- Generates strongly typed data context that is used to send and receive data between the entity classes and the database.
- Data context is the main entry point for LINQ to SQL
- The Data Context is the source of all entities mapped over a database connection. It tracks changes that you made to all retrieved entities and maintains an "identity cache" that guarantees that entities retrieved more than one time are represented by using the same object instance.
- At the same time, the data context implements a number of other common patterns for a domain model: identity map, lazy loading, optimistic offline lock, unit of work, and data mapper.
- O/R designer has two separate panes, entity pane and method pane.
- Drag the tables and views from the server explorer/data explorer on to O/R
- designer in order to create entity class

Scenario:

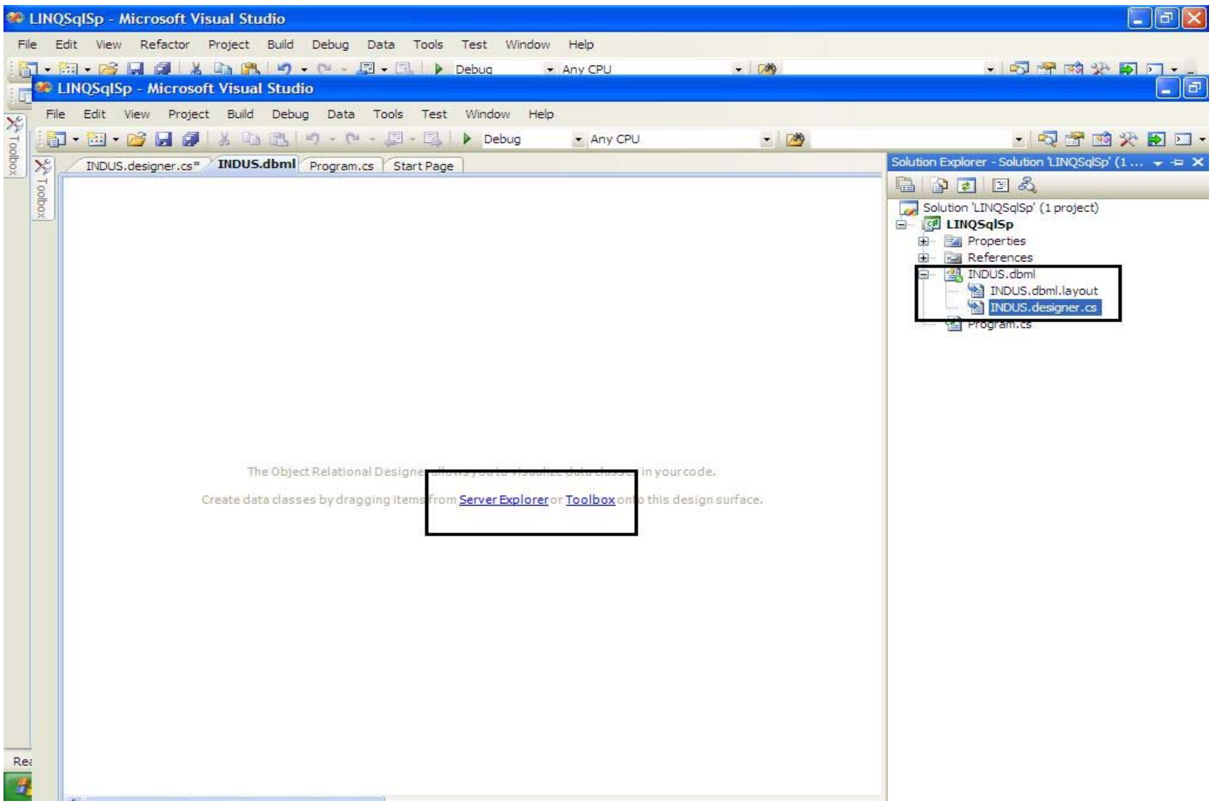
Mr.David, a sales person wants to view the details of all the products that are available in is company. The details of the product reside in products table.Create an entity class for products table using O/R designer.

Demonstration/Code Snippet:

Step 1: Open Visual Studio and create a new Console application. Insert the following Code. It includes the corresponding namespaces.

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

Step 2: Right click on the solution explorer and add new item " LINQ to SQL " classes. Name the file to be "Indus.dbml".Click 'ok' The following two files would be Created under solution explorer.Indus.dbml.layout and Indus.designer.cs.The Indus.dbml would include server explorer as shown in the figure.


LINQSqlSp - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Data Tools Test Window Help
Debug
Any CPU
INDUS.designer.cs*
INDUS.dbml
Program.cs
Start Page
Solution Explorer - Solution 'LINQSqlSp' (1 ...
Solution 'LINQSqlSp' (1 project)
LINQSqlSp
Properties
References
INDUS.dbml
INDUS.dbml.layout
INDUS.designer.cs
Program.cs
Toolbox

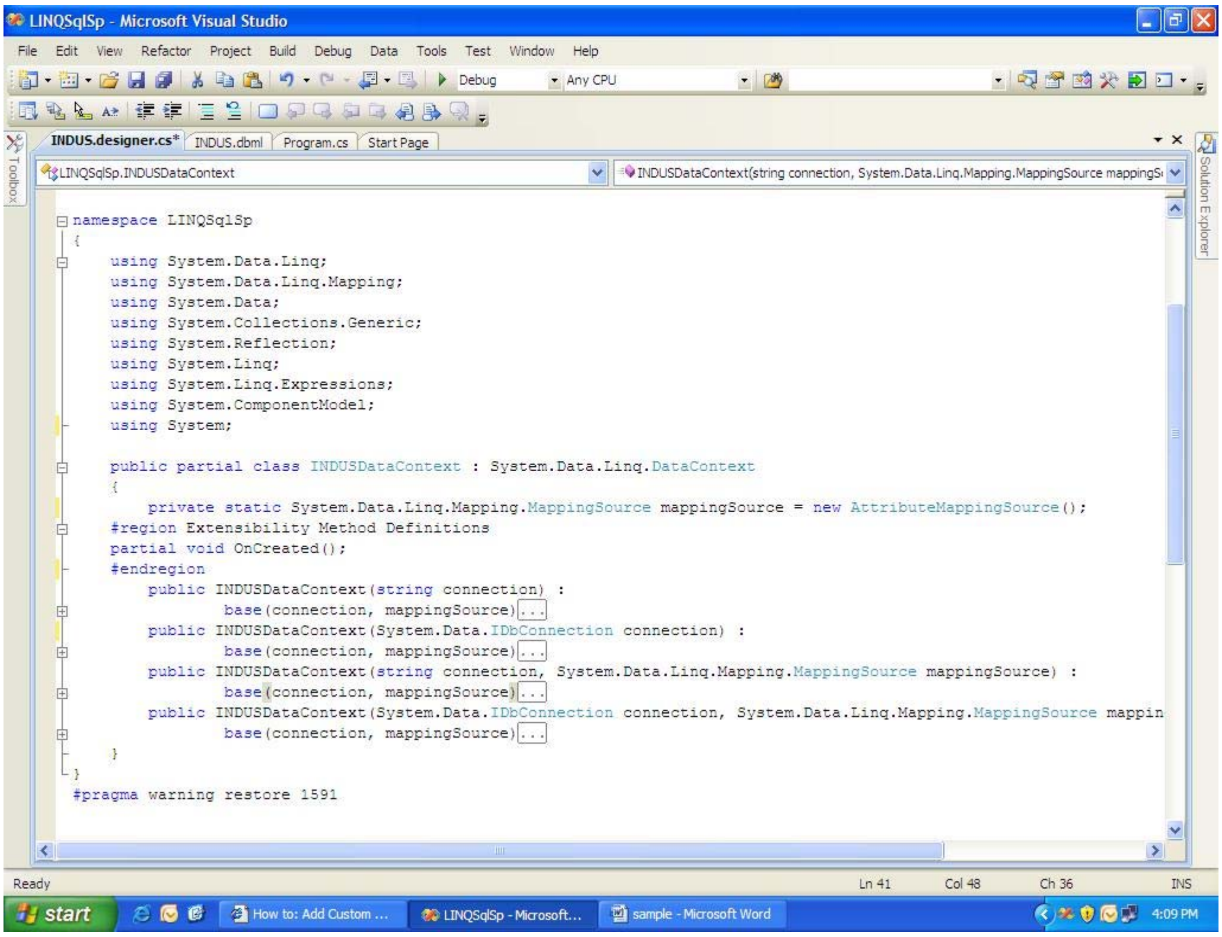
The Object Relational Designer allows you to visualize data classes in your code.

Create data classes by dragging items from Server Explorer or Toolbox onto this design surface.

Figure 2.2-1: Server Explorer on Dbml Layout

Step 3: The Indus.designer.cs shownin the following figure.

.

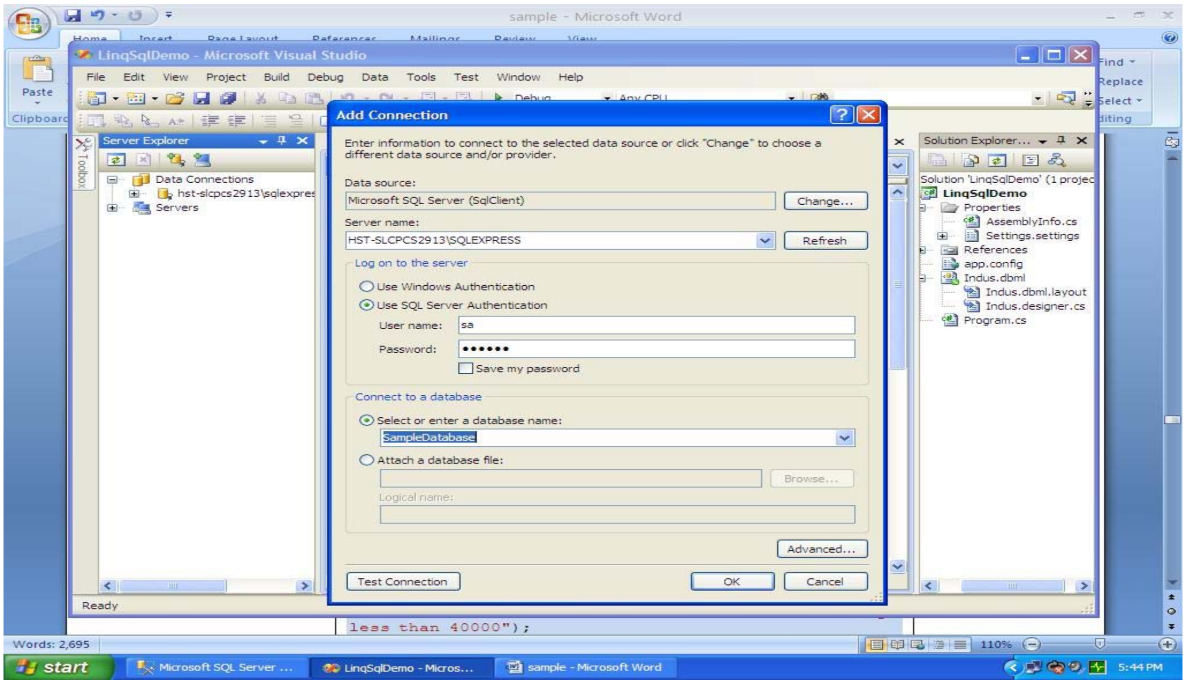

LINQSqlSp - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Data Tools Test Window Help
Debug
Any CPU
INDUS.designer.cs*
INDUS.dbml
Program.cs
Start Page
LINQSqlSp.INDUSDataContext
INDUSDataContext(string connection, System.Data.Linq.Mapping.MappingSource mappingS
Ready
Ln 41
Col 48
Ch 36
INS
start
How to: Add Custom ...
LINQSqlSp - Microsoft...
sample - Microsoft Word
4:09 PM

```
namespace LINQSqlSp
{
    using System.Data.Linq;
    using System.Data.Linq.Mapping;
    using System.Data;
    using System.Collections.Generic;
    using System.Reflection;
    using System.Linq;
    using System.Linq.Expressions;
    using System.ComponentModel;
    using System;

    public partial class INDUSDataContext : System.Data.Linq.DataContext
    {
        private static System.Data.Linq.Mapping.MappingSource mappingSource = new AttributeMappingSource();
    #region Extensibility Method Definitions
    partial void OnCreated();
    #endregion
        public INDUSDataContext(string connection) :
                base(connection, mappingSource)...
        public INDUSDataContext(System.Data.IDbConnection connection) :
                base(connection, mappingSource)...
        public INDUSDataContext(string connection, System.Data.Linq.Mapping.MappingSource mappingSource) :
                base(connection, mappingSource)...
        public INDUSDataContext(System.Data.IDbConnection connection, System.Data.Linq.Mapping.MappingSource mappin
                base(connection, mappingSource)...
    }
}
#pragma warning restore 1591
```

Figure 2.2-2: Indus.Designer.cs

Step 4 : open the server explorer in Indus.dbml layout.Right click on the dataconnections,click Add connection and choose the following as shown in the figure below.



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: HST-SLCPCS2913\SQLEXPRESS

Log on to the server

- Use Windows Authentication
- [✓ Use SQL Server Authentication]

User name: sa

Password: ••••••

Save my password

Connect to a database

- [✓ Select or enter a database name:] SampleDatabase
- Attach a database file:

Logical name:

Figure 2.2-3: Add new Data Connection

Step 5 : Open the new dataconnection in server explorer and drag the required tables on to the layout surface.The entity class is thus created for products table.

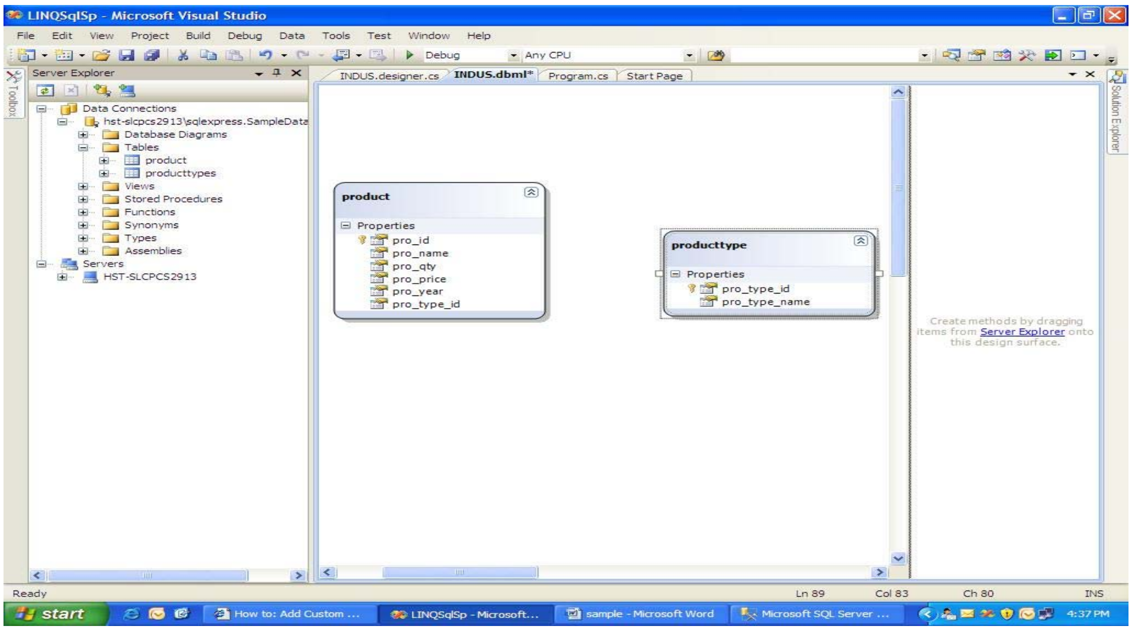
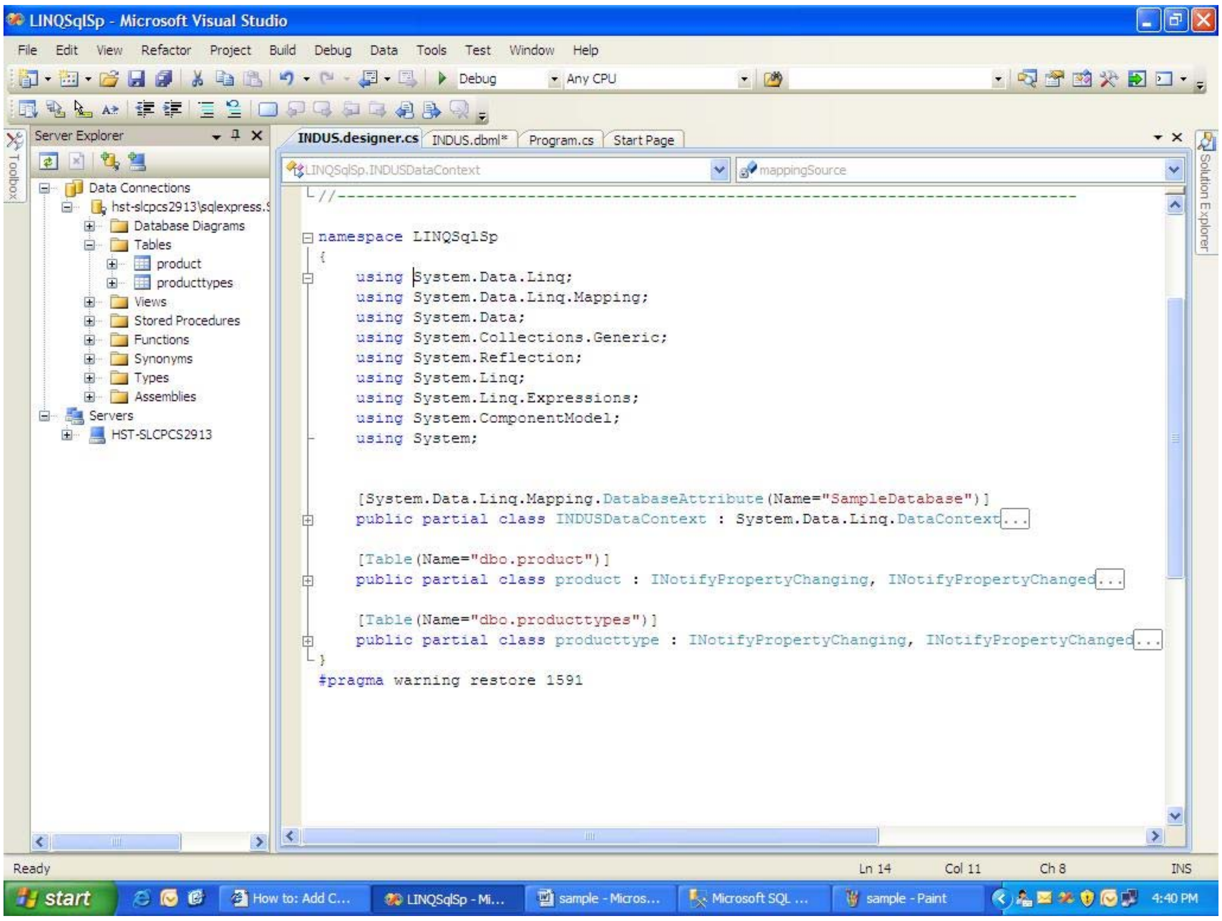


Figure 2.2-4: Indus.Dbml with products table.

Step 5 : The entity class created for products is shown in the following figure



```
//------------------------------------------------------------------------------

namespace LINQSqlSp
{
    using System.Data.Linq;
    using System.Data.Linq.Mapping;
    using System.Data;
    using System.Collections.Generic;
    using System.Reflection;
    using System.Linq;
    using System.Linq.Expressions;
    using System.ComponentModel;
    using System;


    [System.Data.Linq.Mapping.DatabaseAttribute(Name="SampleDatabase")]
    public partial class INDUSDataContext : System.Data.Linq.DataContext...

    [Table(Name="dbo.product")]
    public partial class product : INotifyPropertyChanging, INotifyPropertyChanged...

    [Table(Name="dbo.producttypes")]
    public partial class producttype : INotifyPropertyChanging, INotifyPropertyChanged...
}
#pragma warning restore 1591
```

Figure 2.2-5: Entity class for products table.

Context:

- O/R designer can be used to generate entity classes of LINQ to SQL.

Practice Session:

- Create entity classes for customer and orders tables in north wind database using O/R designer.

Check list:

- Data context methods in O/ R designer and data class hierarchy

Common Errors:

- Unable to connect to the server in order to create a data connection.

Exceptions:

- An exception might occur if we do our coding with respect to a data source which is different from the imported namespace.

Lessons Learnt:

☑ The O/R designer is a simple object relational mapper since it supports 1:1 mapping relationships.

☑ Data Context class produces lightweight and disposable objects that wrap a database connection.

☑ Persisting a data context is not suggested. First, it would eat up a significant share of memory - raw data plus tracking information and mapping information. Second, it is not designed to be a thread-safe object. In general, an instance of the Data Context class should survive a single business operation.

Notes:

- Complex mapping, such as mapping an entity class to a joined table, is not currently supported.
- The designer is a one-way code generator. This means that only changes that you make to the designer surface are reflected in the code file.
- Manual changes to the code file are not reflected in the O/R Designer. Any changes that you make manually in the code file are overwritten when the designer is saved and code is regenerated.

Topic: LINQ-SQLMetal

Estimated Time: 30 min.

Objectives: **At the end of the activity, the participant will be able to understand:**

- Creation of entity classes using SqlMetal.

Presentation:

- SqlMetal Command line tool generates code and mapping for LINQ to Sql component of Dot net framework.
- Useful for generating the C# or VB.NET objects mapped on Sql server database.
- Automatically generates the entity classes for the given database
- The SQLMetal file is included in the Windows SDK that is installed with Visual Studio.
- Default path of the SqlMetal is :
- C: \Program Files\Microsoft SDKs\Windows\Vn.nn\bin
- SqlMetal functionality actually involves two steps:
  - Extracting the metadata of the database into a .dbml file.
  - Generating a code output file.
- The general syntax for using the SqlMetal is

  Sqlmetal [options] [<InputFile>]
- The input file would be a .mdf file,.sdf file or .dbml file
- Sql metal has several options used for creating the entity classes.

Option	**Database**
/server: <name>	Specifies database server name.
/database: <name>	Specifies database catalog on server.
/user: <name>	Specifies logon user id. Default value: Use Windows authentication.
/password: <password>	Specifies logon password. Default value: Use Windows authentication.
/conn: <connection string>	Specifies database connection string.
/timeout: <seconds>	Specifies time-out value when SqlMetal accesses the database.

Figure 2.3-1: Sql Metal Options.

Scenario:

Mr. David works under sales department of a wholesale company. He wants to view the details of all the products available in that company. The details of the product include Product Id, Product name, Product Category; Product Description etc. Use SqlMetal inorder to create entity class for Products table.

Demonstration/Code Snippet:

Step 1: Go to the Command Prompt.

1. Go to the SqlMetal path by typing the following command in the Command Prompt.

```
C: \Program Files\Microsoft SDKs\Windows\V6.0A\bin
```

2.

Context:

- SQL Metal can be used to generate entity classes of LINQ to SQL.

Practice Session:

- Mr. Samuel is the manager of CEIGO medical institute. He wants to view the details of all the doctors available under different departments. Use sqlmetal to create entity classes for staff and departments table.

Check list:

- Output options for sqlmetal

Common Errors:

- Invalid usage of SQL metal commands.
- Proper spacing to be provided for the options.

Exceptions:

- Microsoft SQL Server 2005 throws an exception if one or more of the following conditions are true:
    - SqlMetal tries to extract a stored procedure that calls itself.
    - The nesting level of a stored procedure, function, or view exceeds 32.

Lessons Learnt:

☑ Sqlmetal can perform the following actions

☑ From a database, generate source code and mapping attributes or a mapping file.

☑ From a database, generate an intermediate database markup language (.dbml) file for customization.

☑ From a .dbml file, generate code and mapping attributes or a mapping file.

Best Practices:

- Use sqlmetal for generating the entity classes when you want to query many tables from a database having large number of tables.

Topic: LINQ-SQL Programming

Estimated Time: 30 min.

Objectives: At the end of the activity, the participant will be able to understand:

- LINQ to SQL programming basics.
- Code generation in LINQ to SQL.

Presentation:

- The following steps need to be followed in order to implement the Linq to Sql

  - Create an object model
  - Communicate with the database
  - Create queries (to retrieve\update the database).
  - Submitting the data changes if any.
  - Debugging and testing the application.

- Object Model can be created using either O/R designer or SQL Metal command line tool. In either case, the end-end code generation happens in the following way.

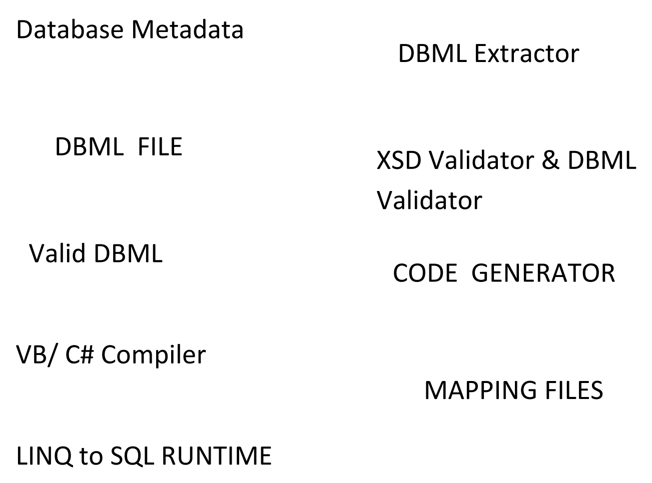
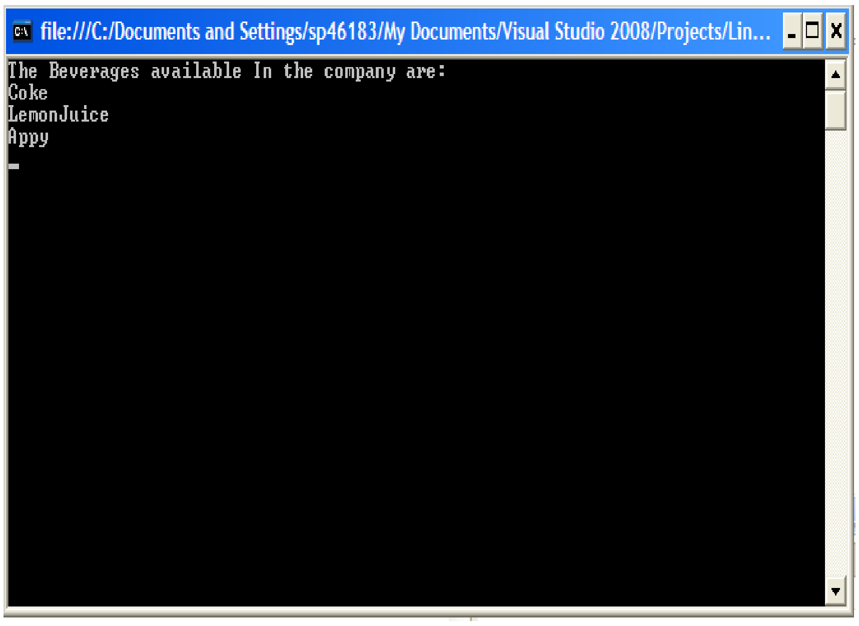


Figure 2.4-1: End-End Code generation

Scenario:

Mr. Samuel is the whole sale seller for INDUS Company. INDUS company maintains products under different categories. He wants to view the details of the products under Beverages category. The details of the products reside inside the “products” table, which include Product Id, Product Name, and Product Category etc. as the columns. Display the list of beverages using LINQ to SQL.

Demonstration/Code Snippet:



file:///C:/Documents and Settings/sp46183/My Documents/Visual Studio 2008/Projects/Lin...

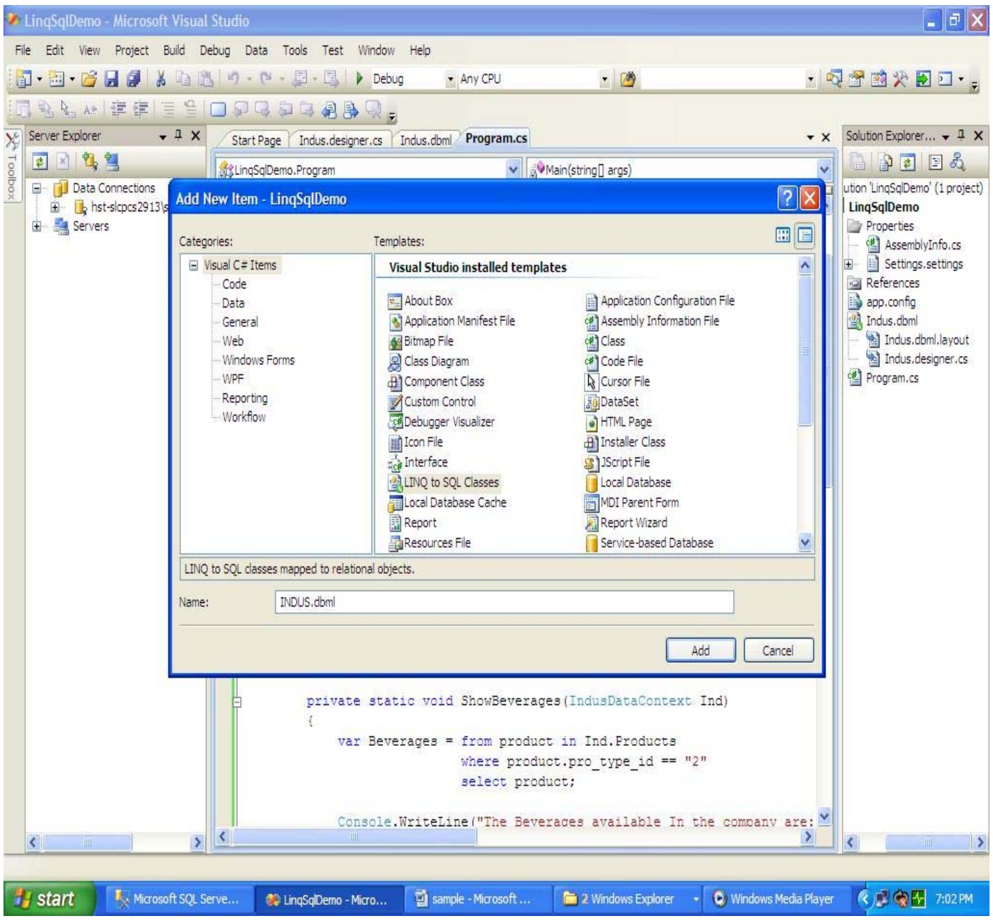
```
The Beverages available In the company are:
Coke
LemonJuice
Appy
```

Figure 2.4-2: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new Console application. Insert the following Code in program.cs. It includes the corresponding namespaces.

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

Step 2: Create an object model using datacontext.Create data context by adding new item”LINQ to SQL” classes. Name the class as Indus.dbml.


LinqSqlDemo - Microsoft Visual Studio
Add New Item - LinqSqlDemo
Categories:
Templates:
Visual Studio installed templates
LINQ to SQL Classes
Add
Cancel

LINQ to SQL classes mapped to relational objects.

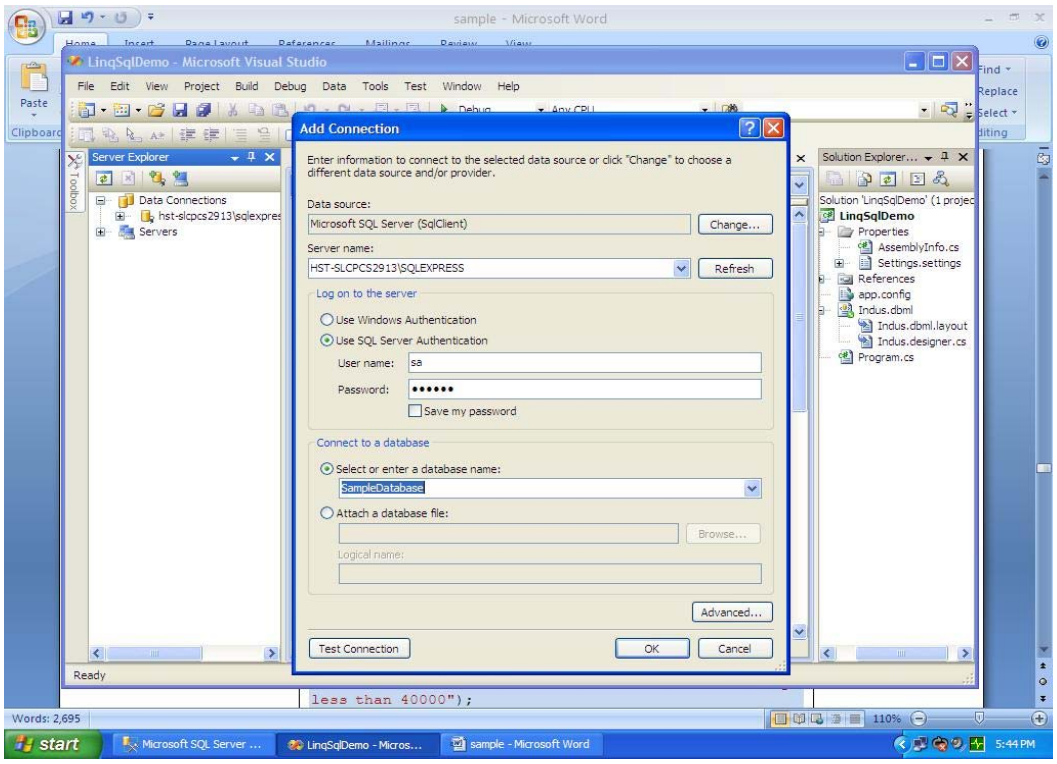
Name: INDUS.dbml

```
private static void ShowBeverages(IndusDataContext Ind)
{
    var Beverages = from product in Ind.Products
                    where product.pro_type_id == "2"
                    select product;

    Console.WriteLine("The Beverages available In the company are:
```

Figure 2.4-3: Output screenshot at the end of execution

Step 3 : Communicate with the database by establising connection to the required database. For creating the connection,open the server explorer in Indus.dbml layout.Right click on the dataconnections,click Add connection and choose the following as shown in the figure below.Drag the products table on to the layout.


Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: HST-SLCPCS2913\SQLEXPRESS

Log on to the server

Use Windows Authentication [✓ Use SQL Server Authentication]

User name: sa

Password: ••••••

Save my password

Connect to a database

[✓ Select or enter a database name:] SampleDatabase

Attach a database file:

Logical name:

Figure 2.4-4: Output screenshot at the end of execution

Step 4 : Querying for displaying the beverages in the company.Insert the following code under program.cs.

```
static void Main(string[] args)
        {
            using (IndusDataContext Ind = new
IndusDataContext())
            {
                var Beverages = from product in
Ind.Products
                                where
product.pro_type_id=="2"
                                select product;
                Console.WriteLine("The Beverages
available In the company are:");

                foreach (Product p in Beverages)
                {
                    Console.WriteLine(p.pro_name);

                }
                Console.ReadLine();
            }

        }
```

Step 5 : To execute the program press “Cntrl + F5” and view the result.

Context:

- To generate Object Model using data context in LINQ to SQL

Practice Session:

- Mr. Paul is the manager of “Indus Medical agency”. He wants to view all the patience details for the year 2009. The details of the products reside inside the “products” table which include Product Id, Product Name, and Product Category etc. as the columns. Display the list of beverages using LINQ to SQL.

Check list:

- When to use data context and its limitations.

Common Errors:

- Invalid server name mentioned while creating a data connection.
- Trying to access a table which is not present in the current data connection.
- Not able to connect to the database.

Exceptions:

- An exception might occur if we do our coding with respect to a data source which is different from the imported namespace.

Lessons Learnt:

☑ DBML (database mark-up language) and its generation.
☑ Implementation of Linq to Sql

Notes:

- The DBML Extractor extracts schema information from the database and reassembles the information into an XML-formatted DBML file.
- The DBML file is scanned by the DBML Validator for errors.
- If no validation errors appear, the file is passed to the Code Generator for mapping files.

.

Topic: LINQ-SQLDMLOperations

Estimated Time: 30 min.

Objectives: **At the end of the activity, the participant will be able to understand:**

- Creation Data Context basics.
- Implementing DML operations using Data context.

Presentation:

- Data context class is a LINQ to SQL class that acts as medium between a SQL server database and LINQ to SQL entities classes mapped to that database.
- Contains the connection string information, methods to connect to the database and manipulating the data in the database.
- Contains several methods that send updated data from LINQ to SQL entity classes to the database.
- Methods can also be mapped to stored procedures and functions.
- With data context, we can perform select, insert, update and delete operations over the data in the database.
- The Core functionality of the data context is explained in the below figure.

`1

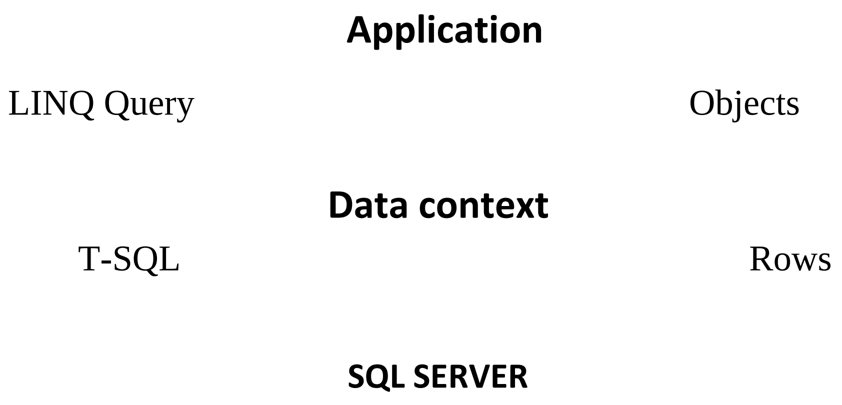
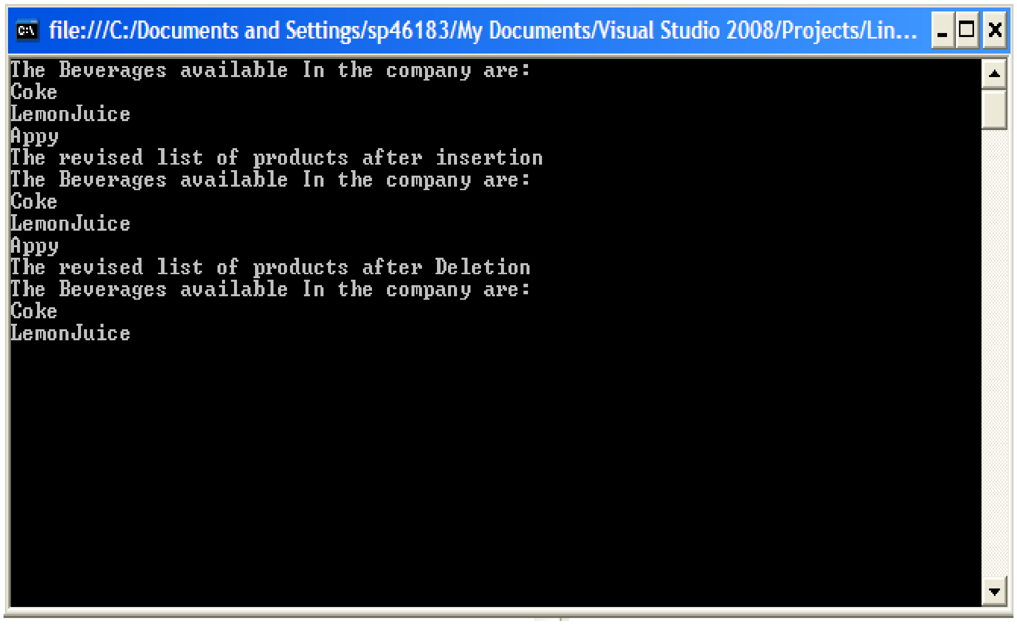


Figure 2.5-1: Query Execution

Scenario:

Mr. Samuel is the whole sale seller for INDUS Company. INDUS company maintains products under different categories. He wants to view the details of the products under Beverages category. The details of the products reside inside the "products" table which include Product Id, Product Name, and Product Category etc. as the columns. He also wants to add new beverage called "Pepsi" and remove "Appy" from the existing list of beverages. Implement this using LINQ to SQL.

Demonstration/Code Snippet:


file:///C:/Documents and Settings/sp46183/My Documents/Visual Studio 2008/Projects/Lin...

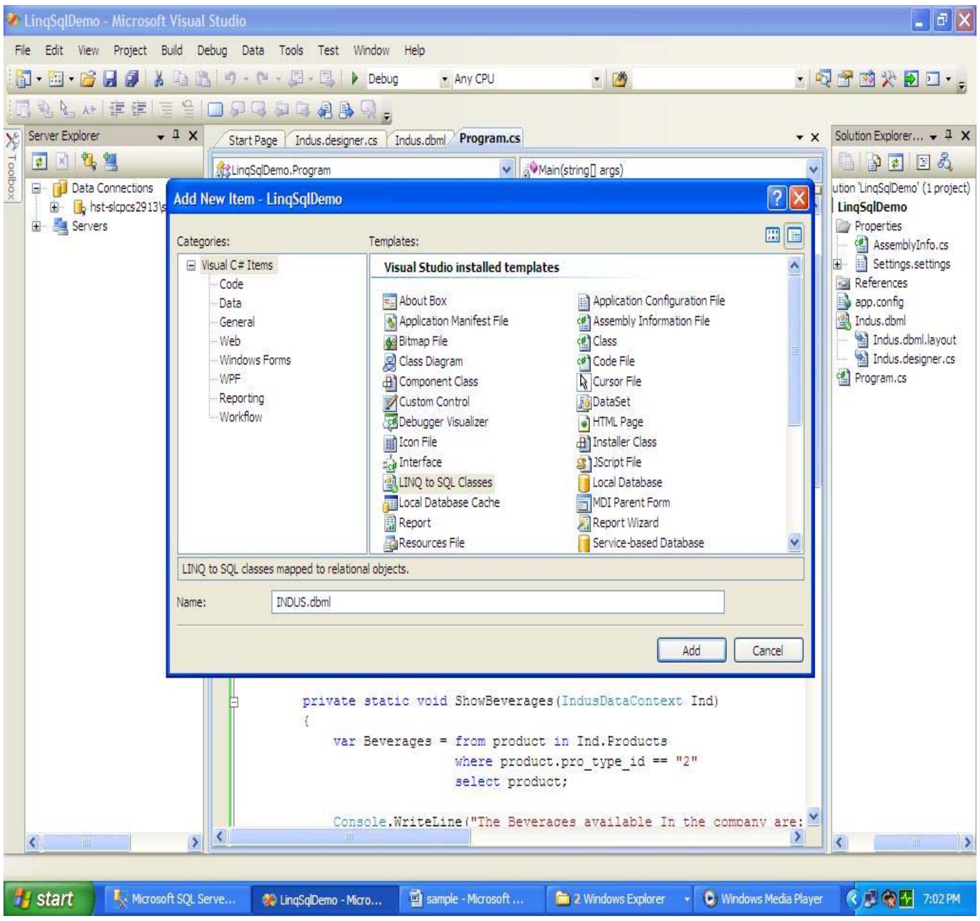
```
The Beverages available In the company are:
Coke
LemonJuice
Appy
The revised list of products after insertion
The Beverages available In the company are:
Coke
LemonJuice
Appy
The revised list of products after Deletion
The Beverages available In the company are:
Coke
LemonJuice
```

Figure 2.5-2: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new Console application. Insert the following Code in program.cs. It includes the corresponding namespaces.

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

Step 2: Create an object model using datacontext.Create data context by adding new item"LINQ to SQL" classes. Name the class as Indus.dbml.



LINQ to SQL classes mapped to relational objects.

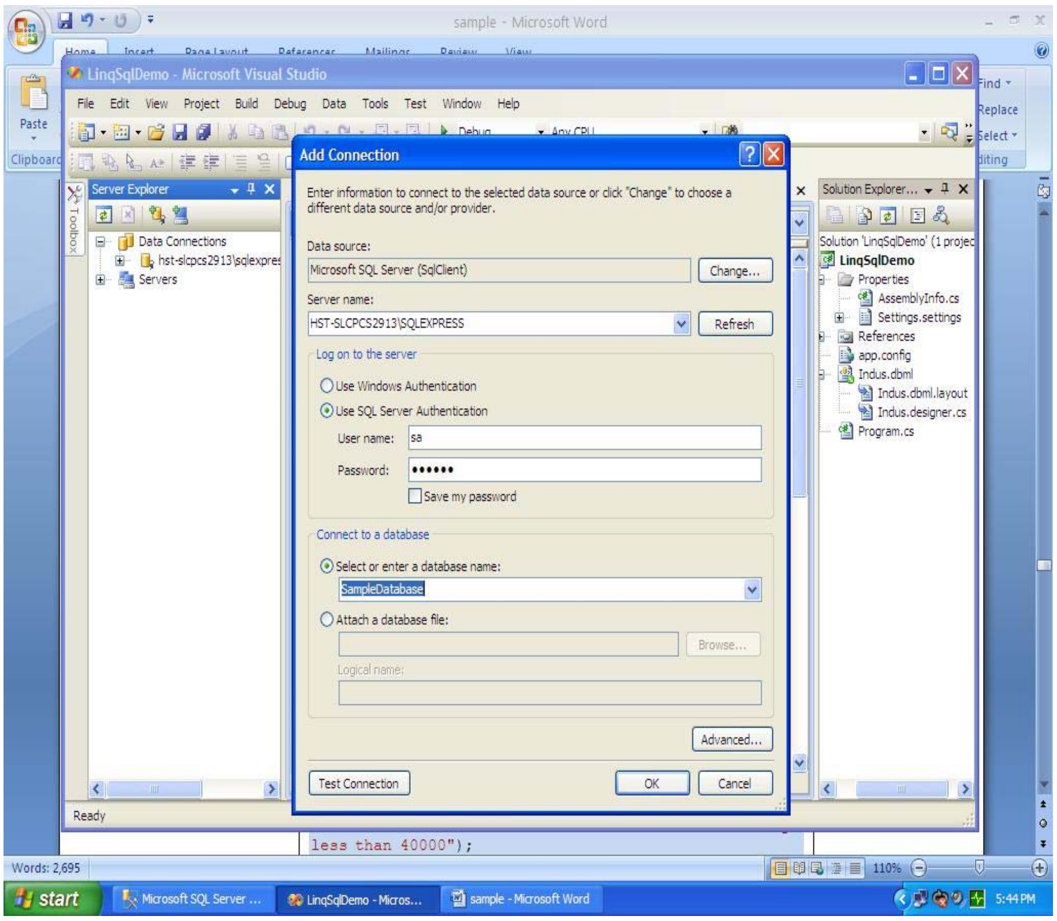
Name: INDUS.dbml

```
private static void ShowBeverages(IndusDataContext Ind)
{
    var Beverages = from product in Ind.Products
                    where product.pro_type_id == "2"
                    select product;

    Console.WriteLine("The Beverages available In the company are:
```

Figure 2.5-3: Output screenshot at the end of execution

Step 3 : Communicate with the database by establising connection to the required database. For creating the connection,open the server explorer in Indus.dbml layout.Right click on the dataconnections,click Add connection and choose the following as shown in the figure below.



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: HST-SLCPCS2913\SQLEXPRESS

Log on to the server

( ) Use Windows Authentication
(•) Use SQL Server Authentication

User name: sa
Password: ••••••
☐ Save my password

Connect to a database

(•) Select or enter a database name: SampleDatabase
( ) Attach a database file:
Logical name:

```
less than 40000");
```

Figure 2.5-4: Output screenshot at the end of execution

Step 4 : Querying for displaying the beverages in the company.Create a method called " Show Beverages" and include the following code

```
private static void ShowBeverages(IndusDataContext Ind)
        {
            var Beverages = from product in Ind.Products
                            where product.pro_type_id == "2"
                            select product;

            Console.WriteLine("The Beverages available In the
company are:");

            foreach (Product p in Beverages)
            {
                Console.WriteLine(p.pro_name);

            }

        }
```

Step5 : Insert new product called “Pepsi” and delete the “Appy” from the existing products.Include the following code under Main method of program.cs

```
static void Main(string[] args)
        {
            using (IndusDataContext Ind = new
IndusDataContext())
            {
                ShowBeverages(Ind);
                Product InsertProduct = new Product { pro_id
= "20", pro_name = "pepsi", pro_qty = 250, pro_price =
35.00m, pro_year = 2009, pro_type_id =" 2 "};
                Ind.Products.InsertOnSubmit(InsertProduct);
                Ind.SubmitChanges();
                Console.WriteLine("The revised list of
                  products after insertion");
                ShowBeverages(Ind);
                Product DeleteProduct = Ind.Products.First(P
                => P.pro_id == "19");
                Ind.Products.DeleteOnSubmit(DeleteProduct);
                Ind.SubmitChanges();
                Console.WriteLine("The revised list of
                products after Deletion");
                ShowBeverages(Ind);
                Console.ReadLine();

            }
```

Step 6: To execute the program press “Cntrl + F5” and view the result.

Context:

- Performing DML operations on data in SQL using Data Context of LINQ to SQL.

Practice Session:

- Mr. Paul is the principal of the “The Model “school. He wants to view the details of the teaching faculty available under different department of his school and also insert the details of vice principal. The details of the faculty include StaffId, StaffName, department, level etc..Implement this using LINQ to SQL.

Check list:

- Different data context methods

Common Errors:

- Syntax errors

Exceptions:

- Trying to update the record after calling submit changes( ) method

Lessons Learnt:

☑ Data context takes some LINQ code and generates appropriate T-SQL statement for that query. Converts the rows returned by the query in to objects.

☑ In order to persist the changes to the SQL server, submit changes () method of data context is to be called after all the changes are done.

Best Practices:

- Use static data context if you want to manipulate the data context at any level.
- Implementation should be easy and non-redundant, so that we do not need to do new Data Context (), every time we have to use one.
- Portability of the Data Context should be such that we do not need to pass it around as parameters from one tier to another.

Mahindra Satyam

Topic: LINQ-SQLStoredProcedures

Estimated Time: 30 min.

Objectives: At the end of the activity, the participant will be able to understand:

- Invoking Stored Procedure using Data context.

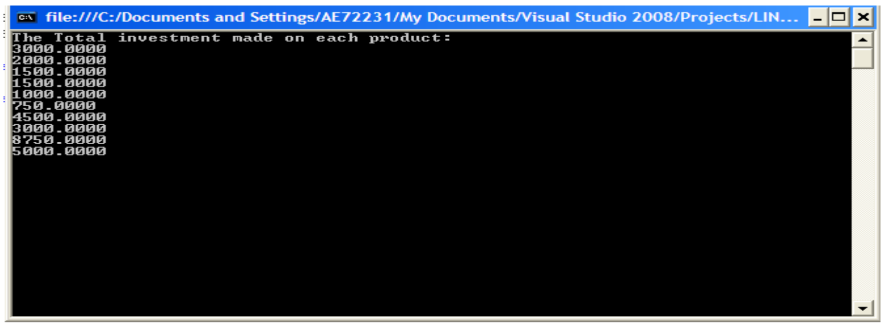
Presentation:

- Stored procedures and functions can be added to O/R designer using Data context methods.
- Calling the method and passing in the required parameters runs the stored procedure or function on the database and returns the data in the return type of the Data Context method
- Data context methods for stored procedures can be created by dragging the stored procedures from server explorer on to the O/R designer.

Scenario:

Mr. Samuel is the whole sale seller for INDUS Company. INDUS Company maintains products under different categories. He wants to view the details of total investment made on buying all the products. The details of the products reside inside the "products" table which includes Product Id, Product Name, and Product Category etc. as the columns. "Total Investment" stored procedure gives the details about it .Implement this using LINQ to SQL.

Demonstration/Code Snippet:


file:///C:/Documents and Settings/AE72231/My Documents/Visual Studio 2008/Projects/LIN...

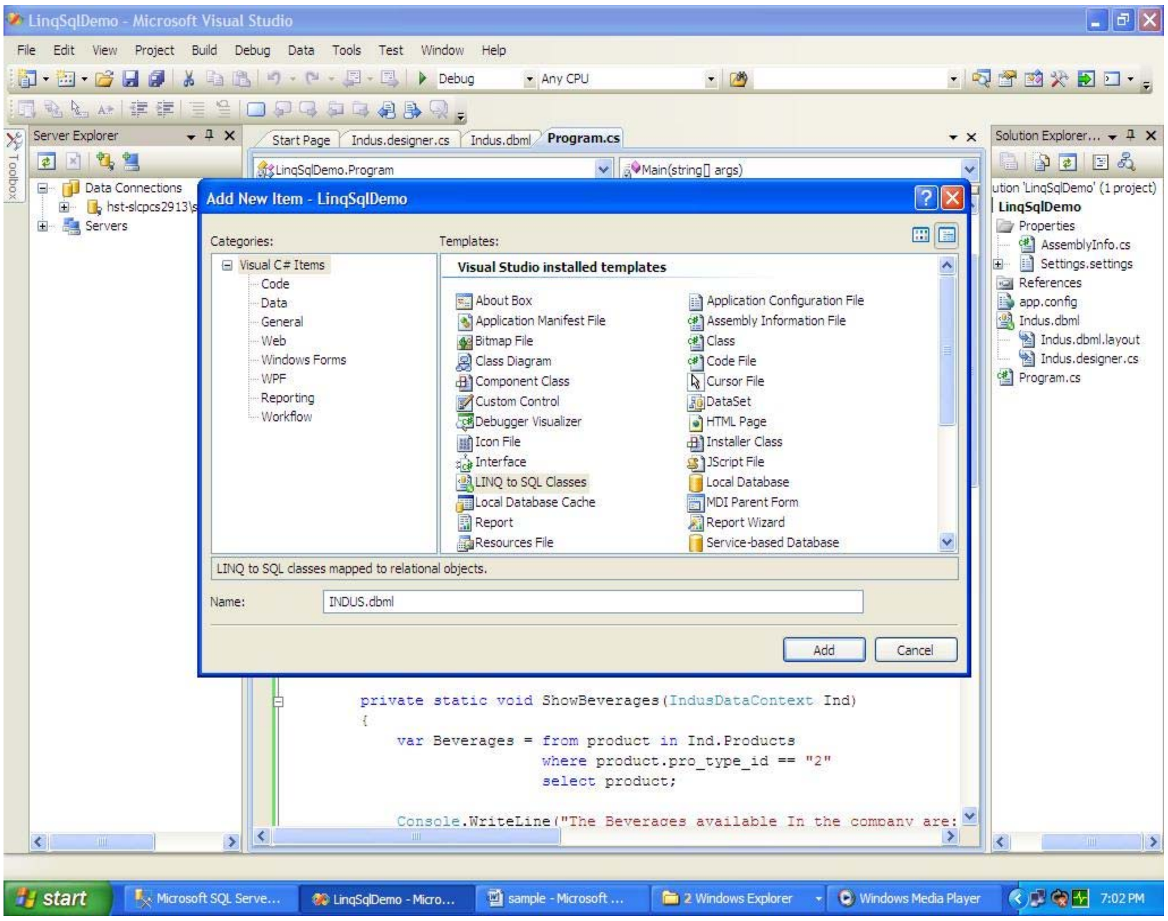
```
The Total investment made on each product:
3000.0000
2000.0000
1500.0000
1500.0000
1000.0000
750.0000
4500.0000
3000.0000
8750.0000
5000.0000
```

Figure 2.6-1: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new Console application. Insert the following Code in program.cs. It includes the corresponding namespaces.

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

Step 2: Create an object model using datacontext.Create data context by adding new item"LINQ to SQL" classes. Name the class as Indus.dbml.



LINQ to SQL classes mapped to relational objects.

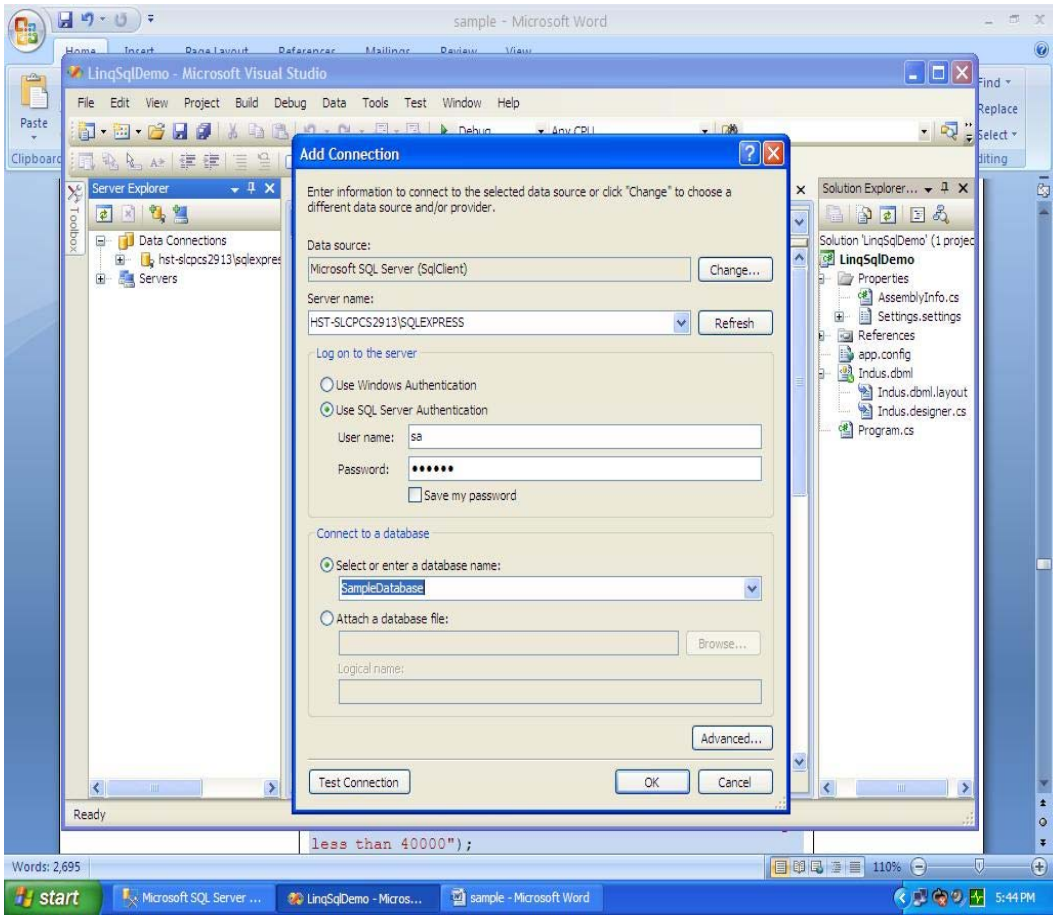
Name: INDUS.dbml

```
        private static void ShowBeverages(IndusDataContext Ind)
        {
            var Beverages = from product in Ind.Products
                            where product.pro_type_id == "2"
                            select product;

            Console.WriteLine("The Beverages available In the company are:
```

Figure 2.6-2: Output screenshot at the end of execution

Step 3 : Communicate with the database by establising connection to the required database. For creating the connection,open the server explorer in Indus.dbml layout.Right click on the dataconnections,click Add connection and choose the following as shown in the figure below.



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: HST-SLCPCS2913\SQLEXPRESS

Log on to the server

( ) Use Windows Authentication
(•) Use SQL Server Authentication

User name: sa

Password: ••••••

[ ] Save my password

Connect to a database

(•) Select or enter a database name: SampleDatabase

( ) Attach a database file:

Logical name:

```
less than 40000");
```

Figure 2.6-3: Output screenshot at the end of execution

Step 4 : Expand the database in the server explorer and drag the required tables on to layout.

Step5 : Expand the database in the server explorer and drag the stored procedure on to the Layout.To view the stored procedure right click on the surface and click"showmethods" pane.

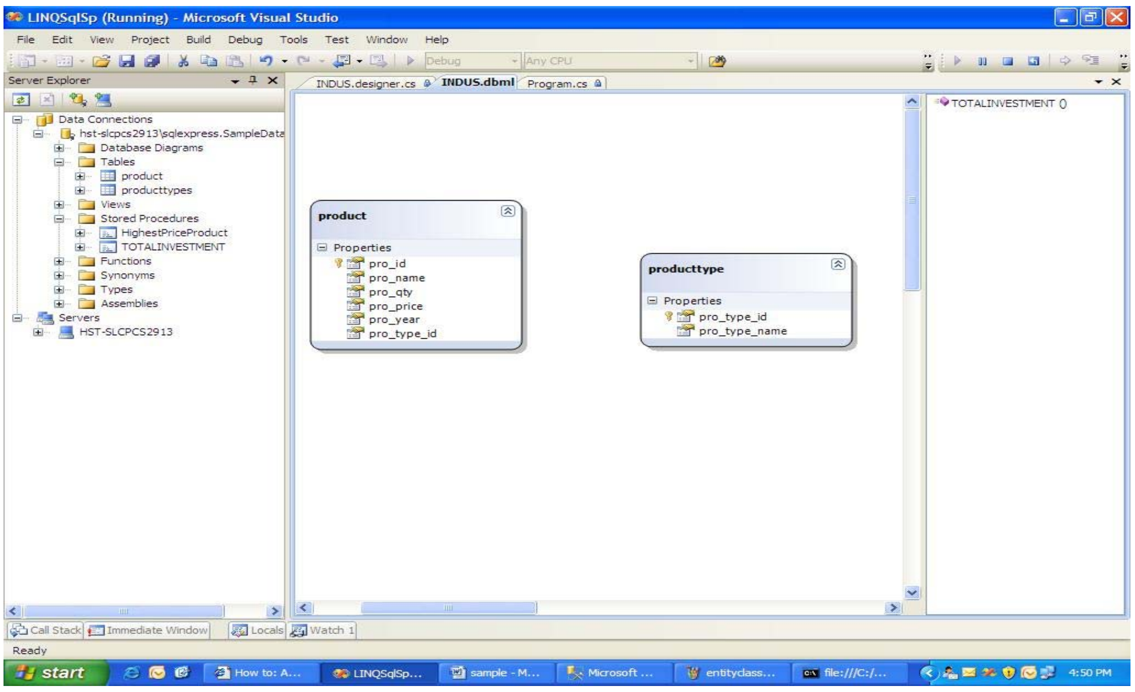


Figure 2.6-4: Output screenshot at the end of execution

Step 6 : Displaying the total investment by invoking the stored procedure. Include the following code under program.cs

```
namespace LINQSqlSp
{
    class Program
    {
        static void Main(string[] args)
        {
            INDUSDataContext IND = new INDUSDataContext();
            Console.WriteLine("The Total investment made on
each product:");
            foreach(var pr in IND.TOTALINVESTMENT())

            {
                Console.WriteLine("{0}", pr.TOTALCOST);
            }
            Console.ReadLine();
        }
    }
}
```

Step 6: To execute the program press “Cntrl + F5” and view the result.

Context:

- Using data context methods to invoke the stored procedures of the sql server.

Practice Session:

- Mrs. Mary is the head of “Globus” institute. She wants to find out the top scorer of the institute for the current year. The stored procedure “calculate aggregate marks” would give the aggregate marks obtained by any student. The stored procedure “Highest scorer” gives the highest value among list of marks provided. Find out and display the top scorer for the Globus institute in a console application using LINQ to SQL.

Check list:

- Different data context methods used while connecting to the database and invoking the stored procedures/functions of the database.

Common Errors:

- Syntax errors
- Not including the namespace of LINQ
- Trying to use stored procedure that is not included under current data context.

Exceptions:

- Trying to access stored procedure that is not part of the current connection of data context.

Lessons Learnt:

☑ Dropping items directly onto an existing entity class creates a Data Context method with the return type of the entity class.

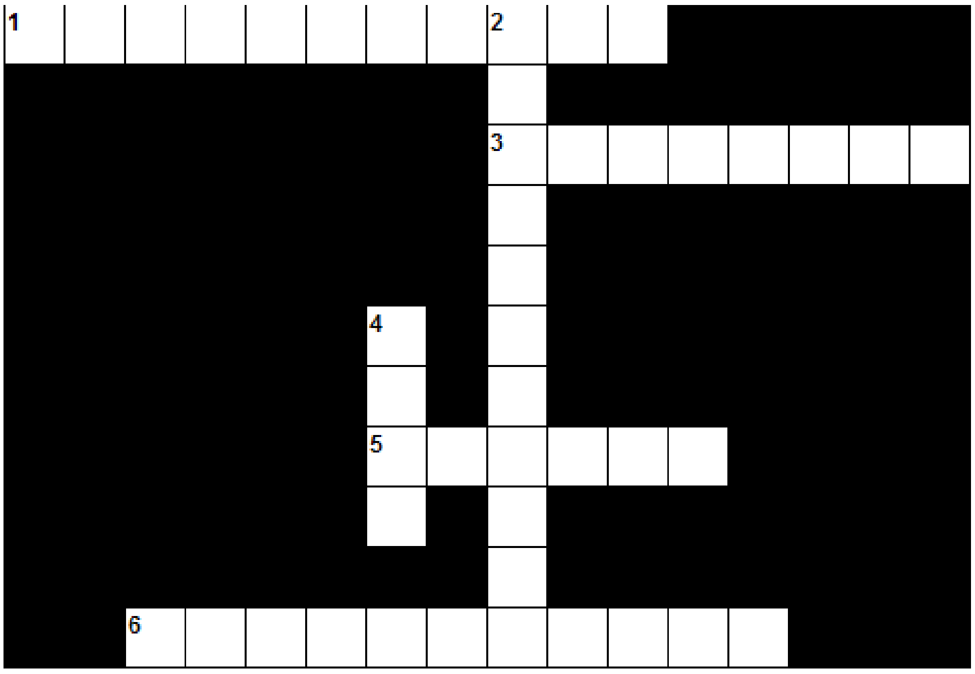
☑ Dropping items onto an empty area of the O/R Designer creates a Data Context method that returns an automatically generated type.

Notes:

- Objects you drag from the database onto the O/R Designer surface will be named automatically, based on the name of the objects in the database.
- If you drag the same object more than once, a number is appended to the end of the new name that differentiates the names
- When database object names contain spaces, or characters not-supported in Visual Basic or C#, the space or invalid character is replaced with an underscore.

Crossword: Unit-2

Estimated Time: 10 min.



Across:

1. Equivalent object for “tables” of relational model(11)

2. Command line tool to create Entity classes(8)

5. The _______ pane of O/R designer contains the stored procedures(6)

6. ________ acts as medium between Sqlserver database and entity classes(11)

Down:

2. Equivalent object for Foreign key relationship of relational model(11)

4. Extension of the entity classes created in LINQ to SQL(4)

.

4.0 LINQ to XML

Topics

Mahindra Satyam

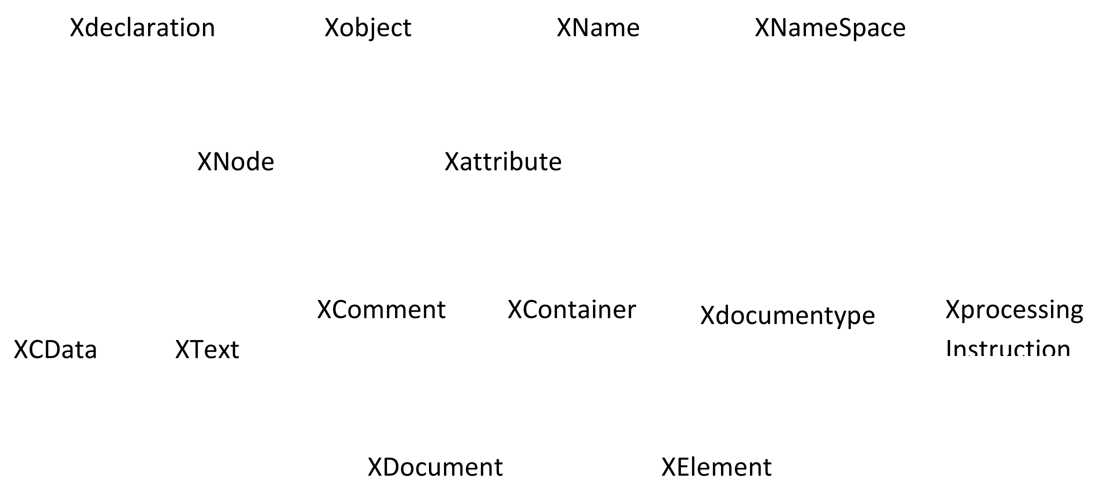
Topic: Introduction to LINQ-XML **Estimated Time: 30 min.**

Objectives: **At the end of the activity ,participants should be able to relate**

- Purpose of LINQ-XML
- LINQ-Xml basics

Presentation:

- LINQ to XML is LINQ-enabled , in-memory XML programming interface
- It is often referred as XLinq.
- LINQ to XML allows XML data to be queried using the standard operators.
- It designed to be light weight Programming API that allows simplifying the programming of XML.
- It provides the in-memory document modification capabilities of the Document Object Model (DOM), and supports LINQ query expres
- Linq to XML uses modern language features like generics and nullable types
- The name space System.Xml.Linq contains many classes for creating and querying XML trees in Xpath style.
- The following diagram represents the LINQ to XML class hierarchy



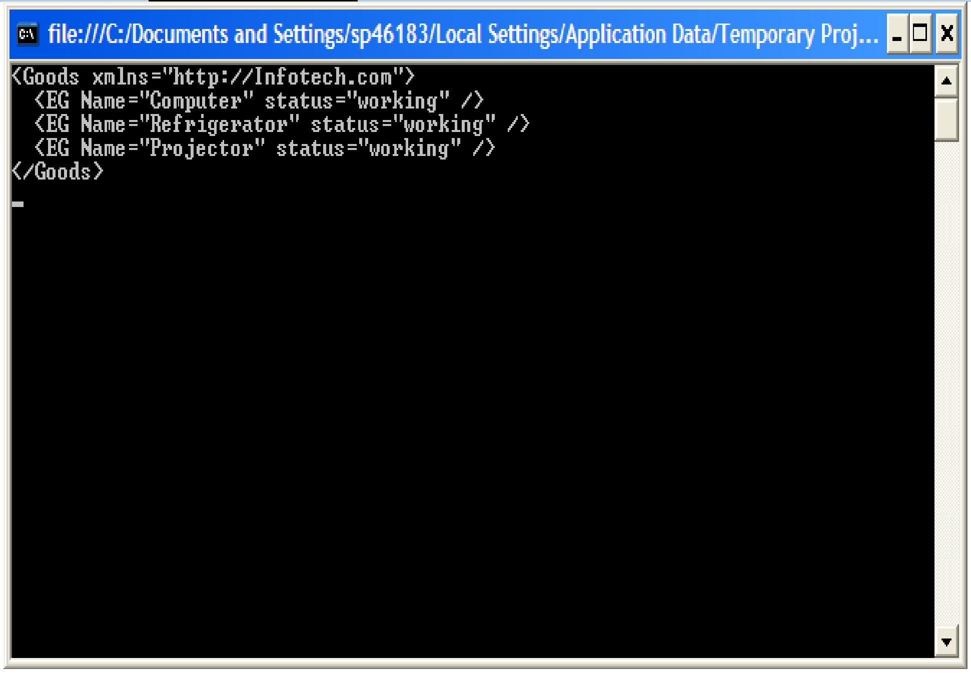
- XName, used for namespace qualified identifies (Qname), used both as element and attribute name

- XNode represents the abstract concept of a node in the XML tree. Acts as a base class for XComment, XContainer, XDocumentType, XProcessingInstruction, and XText.
- XContainer, derived from XNode, offers features such as enumerating children of a node, or finding next and previous sibling; is the base class for XDocument and XElement.
- XDocument represents an XML document, which contains the root level XML constructs, such as: a document type declaration (XDocumentType), one root element (XElement), zero or more comments objects (XComment).
- XElement is the fundamental type for constructing XML data
- XAttribute represents an XML attribute; in other words, a key-value pair associated with an XML element. Attributes are not nodes in the XML tree, thus not derived from XNode.
- XComment, used for comments to the root node or as children of an element.

Scenario:

Mr. Paul is a technician in InfoTech Company. He monitors all the electronic goods of the company. In a weekly report he has to mention the details of all the goods that are in working condition. The details of the goods are represented in Xml. Display the details in a console application using Xlinq.

Demonstration/Code Snippet:


file:///C:/Documents and Settings/sp46183/Local Settings/Application Data/Temporary Proj...

```
<Goods xmlns="http://Infotech.com">
  <EG Name="Computer" status="working" />
  <EG Name="Refrigerator" status="working" />
  <EG Name="Projector" status="working" />
</Goods>
_
```

Figure 4.1-1: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new Console application. Insert the following code. It includes the corresponding namespaces.

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

Step 2: Create Xml with the details using Xlinq Funcitonal construction.

```
XNamespace myComp = "http://Infotech.com";
            XElement Goods = new XElement(myComp +
"Goods",
                                new XElement(myComp +
"EG", new XAttribute("Name", "Computer"), new
XAttribute("status", "working")),
                                new XElement(myComp +
"EG", new XAttribute("Name", "Refrigerator"), new
XAttribute("status", "working")),
                                new XElement(myComp +
"EG", new XAttribute("Name", "Projector"), new
XAttribute("status", "working"))
                                );
```

Step 3 : Display the details in Xml.

```
string Details = Goods.ToString();
Console.WriteLine(Details);
Console.ReadLine();
```

Context:

- To read data and manipulate data from Xml using Linq enabled technology.

Practice Session:

- Mrs. Mary is the owner of a retail outlet. She has to display the details of the products available in her store on daily basis. The details of the products include the name, quantity and price. Implement this using Xlinq and display the results in a console application.

Check list:

- Design principles of LINQ-Xml
- Loading existing Xml

Common Errors:

- Incorrect or insufficient namespaces being imported into the application.
- Syntax errors while writing the query expressions.

Exceptions:

- An exception might occur if we do our coding with respect to a data source which is different from the imported namespace.
- Trying to use invalid expression methods

Lessons Learnt:

☑ The benefit of differed execution is that it allows the same LINQ query to be executed multiple times, giving you the latest results

☑ Sometimes this may lead to performance problems as the query is executed every time.

Notes:

LINQ to XML provides an improved XML programming interface. Using LINQ to Xml, the following can be done

- Load XML from files or streams.
- Serialize XML to files or streams.
- Create XML from scratch by using functional construction.
- Query XML using XPath-like axes.
- Manipulate the in-memory XML tree by using methods such as Add, Remove, ReplaceWith, and SetValue.
- Validate XML trees using XSD.
- Use a combination of these features to transform XML trees from one shape into another.

Topic: Creating Xml using Xlinq

Estimated Time: 30 min.

Objectives: At the end of the activity, the participant should understand

- Creating Xml using Xlinq

Presentation:

- LINQ to XML provides a powerful approach to creating XML elements. This is referred to as functional construction.
- Functional construction allows to create all or part of XML tree in a single statement
- With Linq to Xml it is easy to create Xml trees.
- Creating Xml trees with functional construction is enabled by XElement and XAttribute constructor.
- The XElement uses the following constructor for functional constructor.
  XElement (XName name, object content)
- The content parameter supports any type of object that is a valid child of a XElement.
- For example , it supports the following

  a. A string is added as text content.
  b. A XElement is added as a child element.
  c. A XAttribute is added as an attribute.
  d. A XProcessingInstruction, XComment, or XText is added as child content.
  e. An IEnumerable is enumerated, and these rules are applied recursively to the results.
  f. For any other type, its ToString method is called and the result is added as text content.

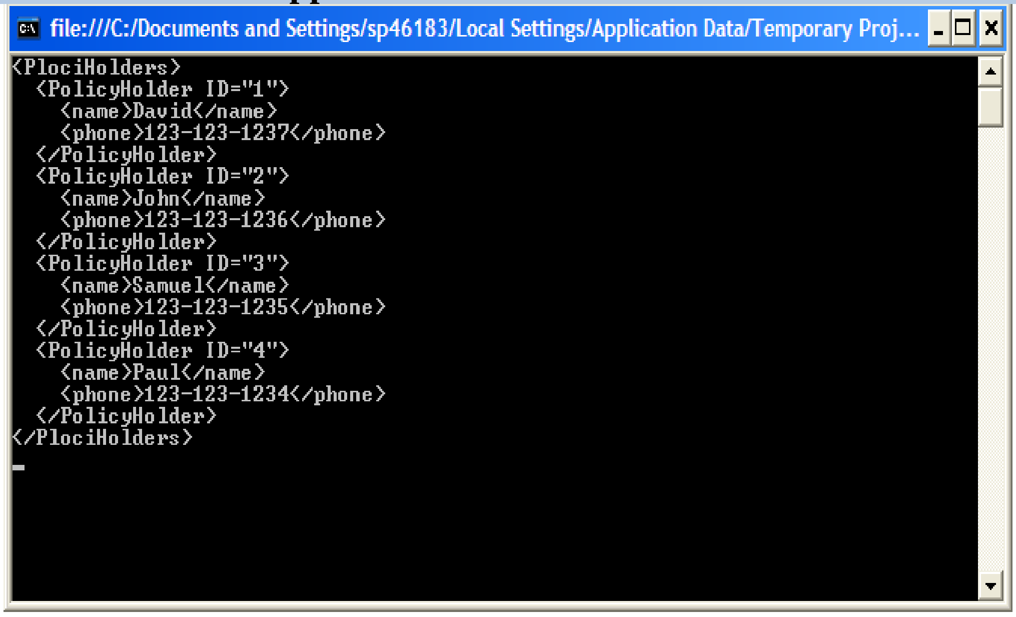
- Using these constructors the Xml trees can be created in the following ways

  a. An empty element
  b. An element with content
  c. An element with Child element
  d. An element with multiple child elements

Scenario:

Mr. John is one of the agents working for ACI, one of the largest insurance companies in India. He wants to view the details of all the policy holders under him. He is also needed to upload the details over the company site. Create the details in Xml using LINQ to Xml and display the details in a console application.

Demonstration/Code Snippet:


file:///C:/Documents and Settings/sp46183/Local Settings/Application Data/Temporary Proj...

```
<PlociHolders>
  <PolicyHolder ID="1">
    <name>David</name>
    <phone>123-123-1237</phone>
  </PolicyHolder>
  <PolicyHolder ID="2">
    <name>John</name>
    <phone>123-123-1236</phone>
  </PolicyHolder>
  <PolicyHolder ID="3">
    <name>Samuel</name>
    <phone>123-123-1235</phone>
  </PolicyHolder>
  <PolicyHolder ID="4">
    <name>Paul</name>
    <phone>123-123-1234</phone>
  </PolicyHolder>
</PlociHolders>
```

Figure 4.2-1: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new Console application. Insert the following code. It includes the corresponding namespaces.

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

Step 2: Create a list with the details of policy holders

```
var ObjPolicyHolders = new[]

              {
           new {ID = 2, Name = "John", Phone = "123-
123-1236"},
           new {ID = 3, Name = "Samuel", Phone =
"123-123-1235"},
           new {ID = 4, Name = "Paul", Phone = "123-
123-1234"},
           new {ID = 1, Name = "David", Phone = "123-
123-1237"}
                };
```

Step 3 : Create Xml using the XElement from the policyholders

```
XElement PolicyHolders = new XElement("PolicyHolders",

                        from c in ObjPolicyHolders
                        orderby c.ID
                        select new
XElement("PolicyHolder",
 new XElement("name", c.Name),
 new XAttribute("ID", c.ID),
 new XElement("phone", c.Phone)  )
                                    );
```

Step 4 : Display the Xml as Output.

```
Console.WriteLine(PolicyHolders);
            Console.ReadLine();
```

Step 5 : Include the Complete Program in Program.cs

```
class Program
    {
        static void Main(string[] args)
        {
            var ObjPolicyHolders = new[]

              {
           new {ID = 2, Name = "John", Phone = "123-123-
1236"},
           new {ID = 3, Name = "Samuel", Phone = "123-
123-1235"},
           new {ID = 4, Name = "Paul", Phone = "123-123-
1234"},
           new {ID = 1, Name = "David", Phone = "123-
123-1237"}
                };
XElement PolicyHolders = new XElement("PlociHolders",

                        from c in ObjPolicyHolders
                        orderby c.ID
                        select new
XElement("PolicyHolder",  new XElement("name", c.Name),
                            new XAttribute("ID", c.ID),
                            new XElement("phone",
c.Phone)  )
                                    );
            Console.WriteLine(PolicyHolders);
            Console.ReadLine();
   }
    }    }
}
```

Context:

- To create the Xml from list using XElement and XAttribute

Practice Session:

- Mr. David is a publisher, preparing journal for “Satyam Computer Services Ltd”. He wants to publish the different Level of the associates maintained in that organization. Details of the associates are maintained in an Xml.Display the details in a console application using Linq to Xml.

Check list:

- Loading existing Xml
- Creating Xml with multiple child elements

Common Errors:

- Incorrect or insufficient namespaces being imported into the application.
- Syntax errors while writing the query expressions.
- Not including System.Xml.Linq namespace.

Exceptions:

- An exception might occur if we do our coding with respect to a data source which is different from the imported namespace.
- Trying to use invalid expression methods

Lessons Learnt:

☑ The benefit of using XElement for creation of XML.
☑ Different ways to create Xml using different constructors of XElement.

Notes:

- When adding XNode (including XElement) or XAttribute objects, if the new content has no parent, the objects are simply attached to the XML tree. If the new content already is parented and is part of another XML tree, the new content is cloned, and the newly cloned content is attached to the XML tree.
- We can save the output Xml to a file, text writer or Xml writer using save method of XElement.

Topic: Querying Xml using Xlinq

Estimated Time: 30 min.

Objectives: At the end of the activity, the participant should understand

- Querying Xml using XLinq

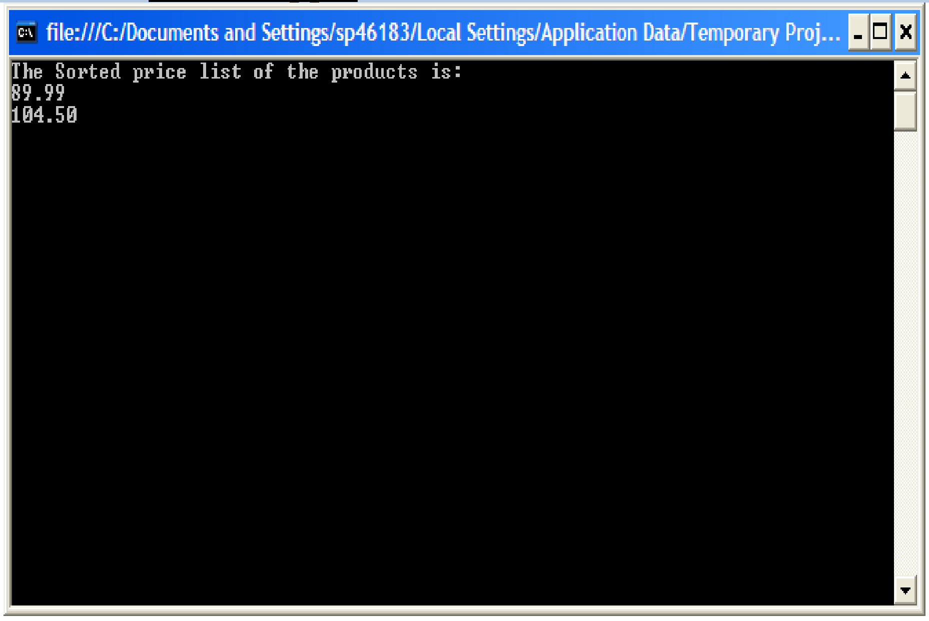
Presentation:

- XLinq greatly reduces the code required to traverse the Xml.
- Querying Xml can be done in the following ways using Xlinq.
  - i. Using Standard Query Operators
  - ii. Using Xml Query extensions
  - iii. Using Xml Transformation
- Xml Query Extensions and Transformation are Xml-Specific querying features provided in Xlinq.
- The standard Query Operator, an API, enables to query all Linq-enabled data sources including Xml.
- There are many standard operators provided by Xlinq under System.Query.Sequence static class.
- Xml transformations including null transformations can be handled very easily using various Standard Query Operators.
- Xml-Specific query operators are generally defined as Extension methods on IEnumerable<XElement>.
- The Xlinq extension methods are defined in XElementSequence class.
- Xml –specific Query extensions provide various Query operations, similar to X-path Axe for working with Xml tree structure.
- Elements, Descendants, Ancestors, Descendants And Self, Ancestors And Self Attributes are some of the Xml Specific Query Extensions.
- Most of the transformations to the xml can be made using the functional construction.
- Any complex functionality can be simplified by incorporating the function calls.

Scenario:

Mr. John is a Wholesale seller. He wants to create a report on the investments done on the products. As part of it , he wants to find the costliest product. Create the details in Xml using LINQ to Xml and display the details in a console application.

Demonstration/Code Snippet:


file:///C:/Documents and Settings/sp46183/Local Settings/Application Data/Temporary Proj...

```
The Sorted price list of the products is:
89.99
104.50
```

Figure 4.3-1: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new Console application. Insert the following code. It includes the corresponding namespaces.

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

Step 2: Consider the Following Xml as input.

```
string parsestr = @"  <Root>
 <Product>
<Category>A</Category>
<Quantity>3</Quantity>
<Price>24.50</Price>
</Product>
<Product>
<Category>B</Category>
<Quantity>1</Quantity>
<Price>89.99</Price>
</Product></Root>";
```

Step 3 : Querying the xml for sorting the prices

```
XElement root = XElement.Parse(@parsestr);
            IEnumerable<decimal> prices =
                from el in root.Elements("Product")
                let price =
(decimal)el.Element("Price")
                orderby price
                select price;
```

Step 4 : Displaying the Output.

```
foreach (decimal el in prices)
                Console.WriteLine(el);
            Console.ReadLine();
```

Step 5 : Include the Complete Program in Program.cs

```
string parsestr = @"  <Root>
<Product>
<Category>A</Category>
<Quantity>3</Quantity>
<Price>104.50</Price>
</Product>
 <Product>
<Category>B</Category>
<Quantity>1</Quantity>
<Price>89.99</Price>
</Product>  </Root>";
  Console.WriteLine("The Sorted price list of the
products is:");
            XElement root = XElement.Parse(@parsestr);
            IEnumerable<decimal> prices =
                from el in root.Elements("Product")
                let price =
(decimal)el.Element("Price")
                orderby price
                select price;
            foreach (decimal el in prices)
                Console.WriteLine(el);
            Console.ReadLine();
```

Context:

- To traverse Xml and retrieve the complete xml or part of it through querying Xml using different query operators.

Practice Session:

- Mr. David is a publisher, preparing journal for “Satyam Computer Services Ltd”. Associates are maintained at different levels in that organization. He wants to publish the names of directors of that company. Details of the associates are maintained in an Xml.Display the details in a console application using Linq to Xml.

Check list:

- Different Xml specific Query operators.
- Advanced Query techniques in Xlinq.
- Xlinq for Xpath Users.

Common Errors:

- Syntax errors while writing the query expressions.
- Not including System.Xml.Linq namespace.

Exceptions:

- Trying to use invalid expression methods

Lessons Learnt:

☑ Creating arbitrarily complex documents in single query using functional construction and ability to incorporate functional calls.
☑ Different ways to traverse Xml.

Best Practices:

- Use the Query Expression syntax wherever possible, it’s easier to read.
- Use XElement when you just need to load the tree of elements, and XDocument when you are interested in any doctype information

Notes:

- The Descendants and Ancestors query operators let you query down and up the XML tree, respectively
- Descendants with no parameters gives you all the child content of an XElement
- Descendants and Ancestors do not include the current node.
- The Attributes XML query extension is called on an IEnumerable<XElement> and returns a sequence of attributes (IEnumerable<XAttribute>).
- The ElementsBeforeSelf query extension returns an IEnumerable<XElement> containing the sibling elements that occur before that element.
- Elements after self returns the sibling elements that occur after that element.
- The NodesBeforeSelf query extension returns the previous siblings of any type. for example, string, XComment, XElement, and so on

Topic: Manipulating Xml using Xlinq

Estimated Time: 30 min.

Objectives: At the end of the activity, the participant should understand

- Inserting Xml using Xlinq
- Updating Xml using Xlinq
- Deleting Xml using Xlinq

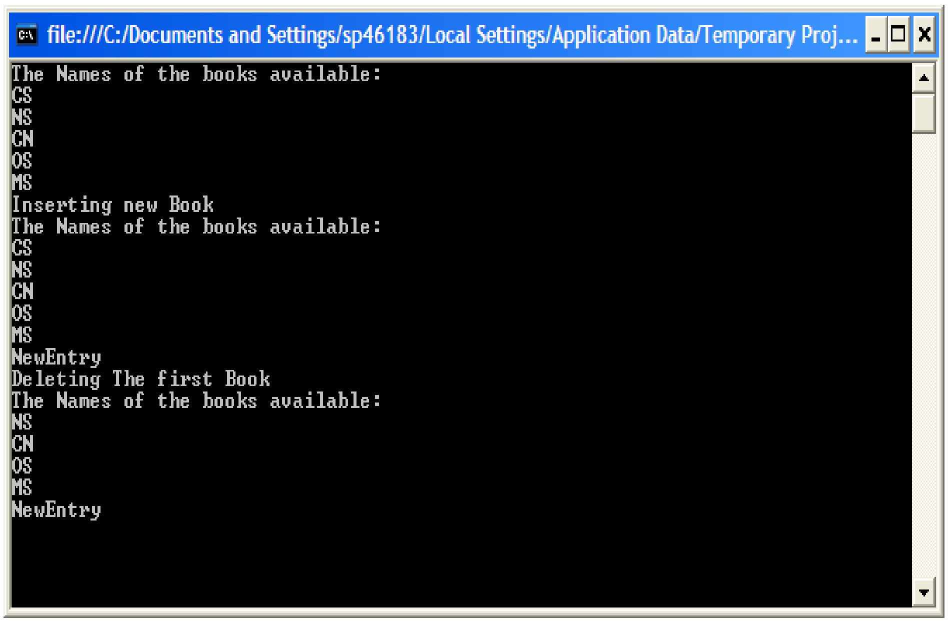
Presentation:

- XLinq provides a full set of methods for manipulating XML.
- The following can be performed on Xml using Xlinq
  - Inserting Xml using Xlinq
  - Updating Xml using Xlinq
  - Deleting Xml using Xlinq
- The Add method is used to add content to the existing Xml.The Add methods work similarly to the XElement and XDocument so that full XML subtrees can be added using the functional construction style.
- AddFirst() is used if the content is to be added to the beginning of the children
- AddBeforeSelf() or AddAfterSelf() can be used to navigate to a child before or after the target location
- Remove () is used for deleting an element from Xml after navigating to the content that is to be deleted.
- RemoveNodes() method can be used for removing all the content from Xml.
- Another way to remove an element is to set it to null using SetElement.
- For updating the Xml, navigate to the content that needs to be replaced and use ReplaceNodes () method.
- ReplaceContent (), Setelement () can also be used for updating the Xml sub tree.

Scenario:

Mr. Samuel is the owner of a bookstore.He maintains the details of all the books available in his store. He monitors the list of books on regular basis by adding new entries and deleting the old ones. Implement this using Xlinq and display the results in console application.

Demonstration/Code Snippet:


file:///C:/Documents and Settings/sp46183/Local Settings/Application Data/Temporary Proj...

```
The Names of the books available:
CS
NS
CN
OS
MS
Inserting new Book
The Names of the books available:
CS
NS
CN
OS
MS
NewEntry
Deleting The first Book
The Names of the books available:
NS
CN
OS
MS
NewEntry
```

Figure 4.4-1: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new Console application. Insert the following code. It includes the corresponding namespaces.

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

Step 2: Consider the Following Xml as input.

```
string Inputstr = @"<Books>
<Book><BNo>11</BNo><Title>CS</Title></Book>
<Book><BNo>12</BNo><Title>NS</Title></Book>
<Book><BNo>13</BNo><Title>CN</Title></Book>
<Book><BNo>14</BNo><Title>OS</Title></Book>
<Book><BNo>15</BNo><Title>MS</Title></Book>
</Books>";
```

Step 3 :Method for displaying the details of the books.

```
private static void DisplayBooks(XElement books)
        {
            var Details = from b in books.Elements()
                          let Title =
(string)b.Element("Title")
                          select Title;

            Console.WriteLine("The Names of the books
available:");
            foreach (string bookDetails in Details)
            {

                Console.WriteLine(bookDetails);
            }
        }
```

Step 4 : Insert a new book using Add method

```
Console.WriteLine("Inserting new Book");
            XElement newBook = new XElement("Book",
new XElement("BNo", 16),
                                              new
XElement("Title", "NewEntry"));
            books.Add(newBook);
            DisplayBooks(books);
```

Step 5 : Delete the first book using Remove

```
Console.WriteLine("Deleting The first Book");
            books.Element("Book").Remove();
            DisplayBooks(books);
```

Step 6 : Include the complete program under Mainmethod of program.cs

```
string Inputstr = @"<Books>

<Book><BNo>11</BNo><Title>CS</Title></Book>
<Book><BNo>12</BNo><Title>NS</Title></Book>
<Book><BNo>13</BNo><Title>CN</Title></Book>
<Book><BNo>14</BNo><Title>OS</Title></Book>
<Book><BNo>15</BNo><Title>MS</Title></Book>
</Books>";
 XElement books = XElement.Parse(Inputstr);
 DisplayBooks(books);
  Console.WriteLine("Inserting new Book");
XElement newBook = new XElement("Book", new
XElement("BNo", 16),
new XElement("Title", "NewEntry"));
            books.Add(newBook);
            DisplayBooks(books);
Console.WriteLine("Deleting The first Book");
            books.Element("Book").Remove();
            DisplayBooks(books);
            Console.ReadLine();
```

Context:

- To manipulate the Xml either by adding, updating or deleting the content.

Practice Session:

- Mr. David is the owner of SmartTech Company. The employee's details are monitored and updated (adding new employees and deleting the details of employees who have left) on regular basis. Implement this using Xlinq and display the results in Console application.

Check list:

- ReplaceContent and ReplaceNode methods
- Updating only sub element
- Updating the Entire element.

Common Errors:

- Syntax errors while writing the query expressions.
- Not including System.Xml.Linq namespace.

Exceptions:

- Trying to use invalid expression methods

Lessons Learnt:

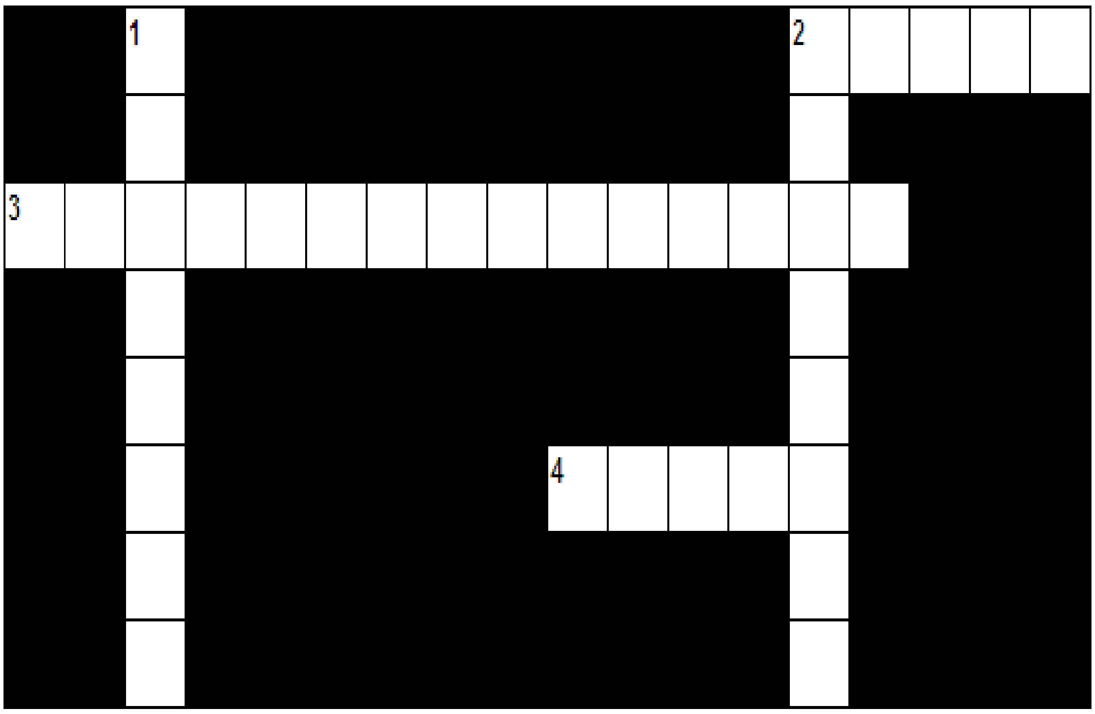
- ☑ While adding to xml, if the child is already parented, LINQ to XML clones the child element under subsequent parents.
- ☑ The query extension Remove () is one of the few extension methods that does not use deferred execution and uses exactly this ToList () approach to cache up the items targeted for deletion.

Best Practices:

- Use the Query Expression syntax wherever possible, it's easier to read.
- XElement and XDocument are similar, but not interchangeable when you are loading a document from a data source. Use XElement.Load () when you want to load everything under the top-level element, Use XDocument.Load () when you want to load any markup before the top-level element.

Crossword: Unit-4

Estimated Time: 10 min.



ACROSS:

2. LINQ to XMl is also called as _____(5).
3. The _____________ query extension returns the previous siblings of any XElement(15)
4. ________ represents the abstract concept of a node in the XML tree(5).

DOWN:

1. Method which is used if the content is to be added to the beginning of the children (8).
2. Descendants with no parameters gives you all the child content of ___________(8).

4.0 LINQ to DATASET

Topics

Topic: LINQ TO Dataset

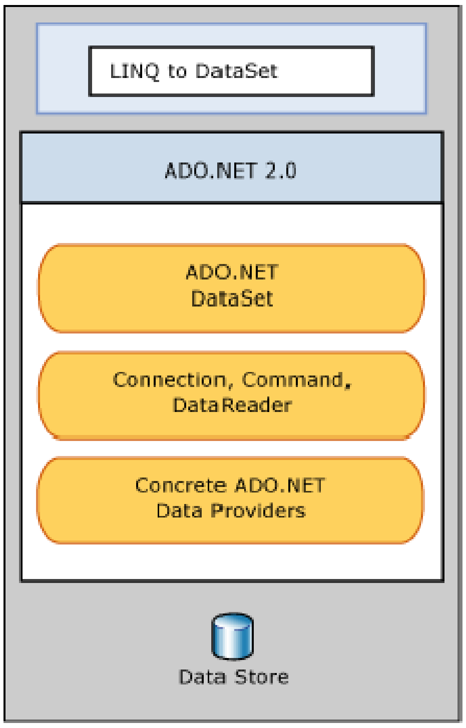
Estimated Time: 30 min.

Objectives: At the end of the activity, the participant should understand

- LINQ to dataset basics
- Working with LINQ to dataset

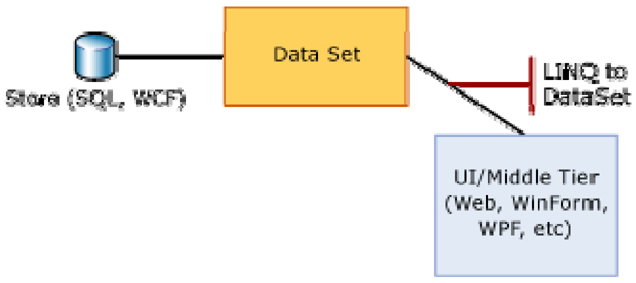
Presentation:

- LINQ to DataSet makes it easier and faster to query over data cached in a DataSet object
- LINQ to DataSet can also be used to query over data that has been consolidated from one or more data sources
- The LINQ to DataSet functionality is exposed primarily through the extension methods in the DataRowExtensions and DataTableExtensions classes.
- LINQ to DataSet builds on and uses the existing ADO.NET 2.0 architecture, and is not meant to replace ADO.NET 2.0 in application code.


LINQ to DataSet
ADO.NET 2.0
ADO.NET DataSet
Connection, Command, DataReader
Concrete ADO.NET Data Providers
Data Store

-
-

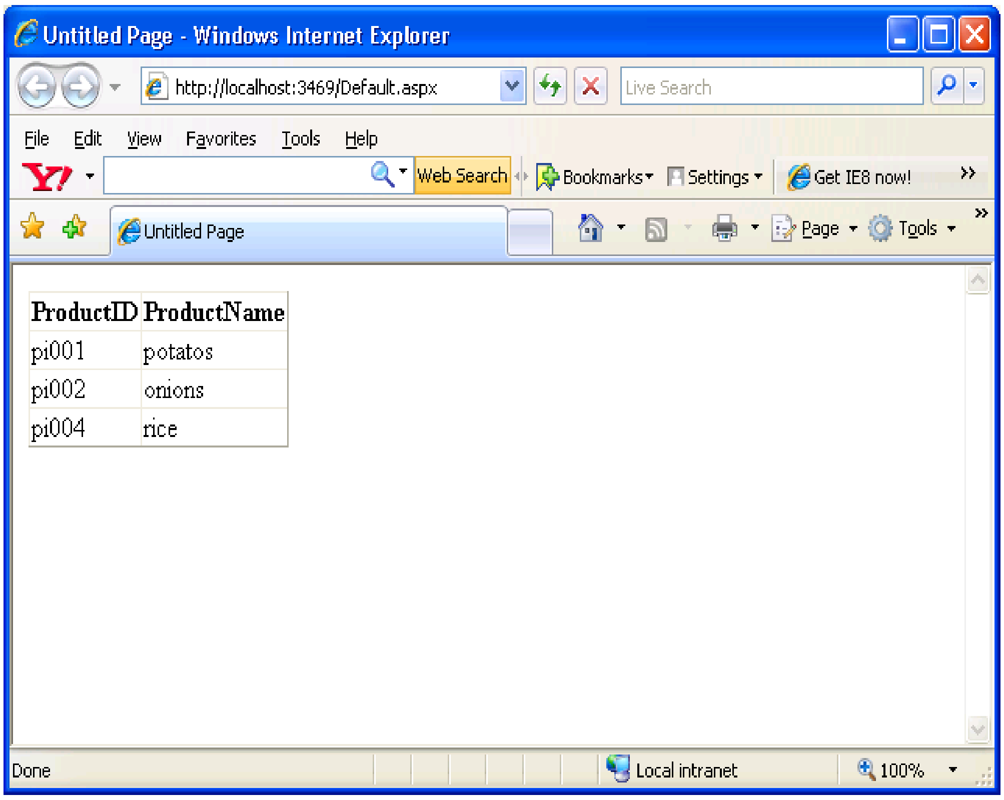
- LINQ queries are expressed in the programming language itself, rather than as string literals embedded in the application code. This means that developers do not have to learn a separate query language.
- LINQ to DataSet enables Visual Studio developers to work more productively, because the Visual Studio IDE provides compile-time syntax checking, static typing, and IntelliSense support for LINQ.
- There are several ways of loading data into DataSet, using a DataAdapter class, LINQ to SQL, wizard.
- Below is a presentation of how LINQ to DataSet fits in an N-Tier Architecture Applications.



Scenario:

- ABC is a Departmental Store, maintaining its database with all the details of the products it deals with. One of its managers wants to view the availability of the stock at a particular warehouse to place an order to the company. Create an application that displays the products that are purchased in year 2009.

Demonstration/Code Snippet:


Untitled Page - Windows Internet Explorer
http://localhost:3469/Default.aspx
Live Search
File Edit View Favorites Tools Help
Web Search
Bookmarks
Settings
Get IE8 now!
Untitled Page
Page
Tools
Done
Local intranet
100%

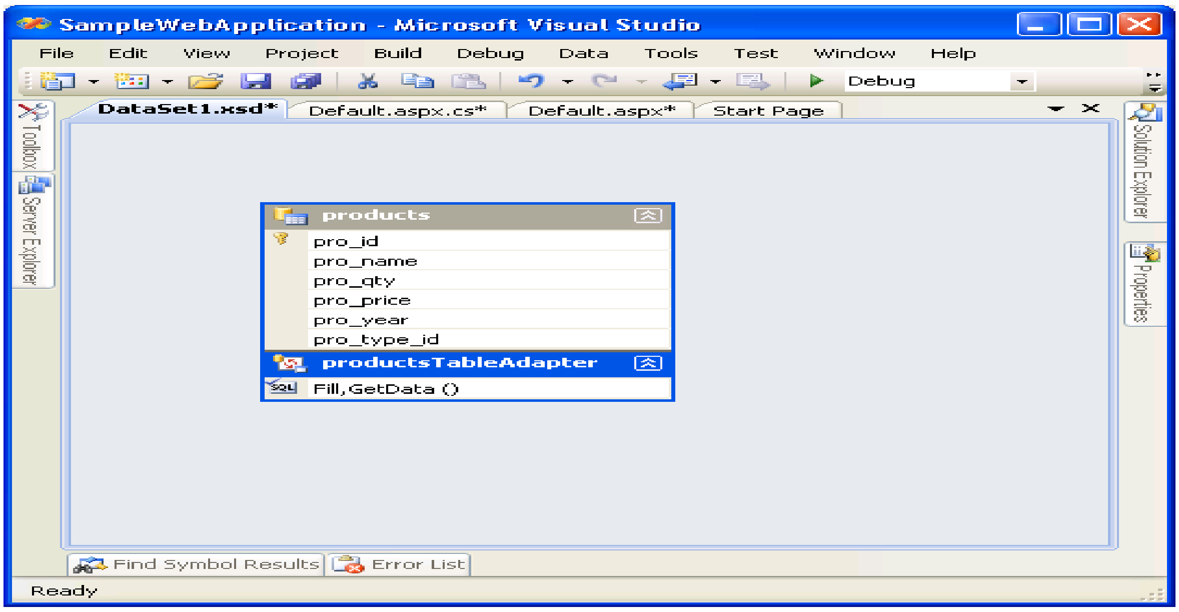
ProductID	ProductName
pi001	potatos
pi002	onions
pi004	rice

Figure 5.1-1: Output screenshot at the end of execution

Step 1: Open Visual Studio and create a new web application.
Step 2: Consider the Following Xml as input.

```
string Inputstr = @"<Books>
<Book><BNo>11</BNo><Title>CS</Title></Book>
<Book><BNo>12</BNo><Title>NS</Title></Book>
<Book><BNo>13</BNo><Title>CN</Title></Book>
<Book><BNo>14</BNo><Title>OS</Title></Book>
<Book><BNo>15</BNo><Title>MS</Title></Book>
</Books>";
```

Step 3: Go to Solution Explorer, right click on your project and go to add, new item, click on Data on the left pane, and select DataSet1.xsd.
Step 4 : Go to the DataSet1.xsd file and go to ServerExplorer, connect to the server and select the appropriate database.
Step 5 : From the ServerExplorer drag the Products tables to the DataSet.xsd window.



Step 6 : Go to Default.aspx page and place a GridView from the Toolbox.
Step 7 : Go to Default.aspx.cs page and place the below code.

```
namespace SampleWebApplication
{
    public partial class _Default : System.Web.UI.Page
    {
     protected void Page_Load(object sender, EventArgs e)
        {

            //Create a prodctsTableAdapter object
            //This Table Adapter come as a default with
the table/view we select from the data scource into the
DataSet.xsd file.
            //Table Adapter has two default methods fill
and get.
            DataSet1TableAdapters.productsTableAdapter
dt1 = new DataSet1TableAdapters.productsTableAdapter();

            var query = from p in dt1.GetData()
                        where p.pro_year == 2009
                        select new
                        {
                            ProductID = p.pro_id,
                            ProductName = p.pro_name
                        };


            GridView1.DataSource = query;
            GridView1.DataBind();

        }
    }
```

Context:

- Creating Datasets through wizards and performing LINQ Operations.

Practice Session:

- ABC wants to view the details of the products which have an expiry date as per month and year. Try creating a stored proc which takes month and year as the input, and use this proc in the project.

Check list:

- Limitations of LINQ to Dataset

Common Errors:

- Not including correct namespace.
- Incorrect information being provided in the connection string or query string.

Exceptions:

- Trying to use invalid query operators

Lessons Learnt:

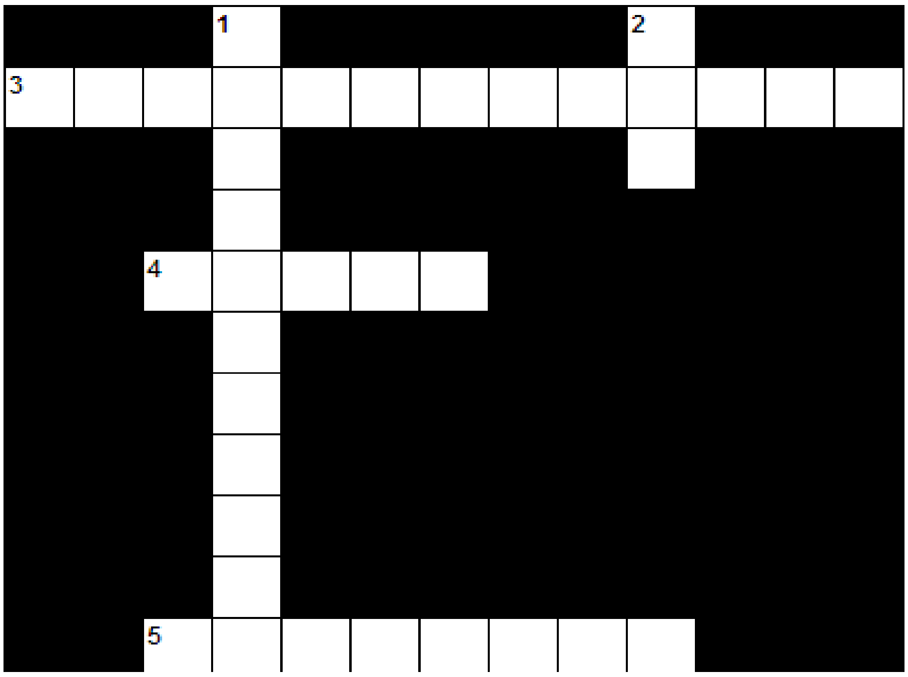
☑ Use of DataSet, TableAdapters, and LINQ operations

Notes:

- LINQ to DataSet allows us to make use of the new superior LINQ querying features in legacy applications without having to rewrite the whole data access layer
- A typed DataSet is preferable, because it can be queried with a simpler syntax because it is not necessary to use the Field<T> accessor and the AsEnumerable method

Crossword: Unit-5

Estimated Time: 10 min.



Across:

1. ______________ generate classes that expose each object the in the DataSet in Type-safe manner.(13)
2. ___________ Adapters provide communication between your application and a database by executing SQL statements and stored procedures against a database.(5)
3. A query that is not evaluated when it is created or declared, but only when the query is executed, called as ______________ execution.(8)

Down:

1. Operators such as DISTINCT, UNION, INTERSECT, & EXCEPT, compare source elements by calling the _____________.(11)
2. ____ is the extension of the item (dataset) which you create and include it in your project.(3)

Answers for Crosswords

Unit-1

Across	**1)IQueryable 3)Deferred 5)Expressiontree 6)Var**
Down	**2)Skip 4)Empty**

Unit-2

Across	**5)Method 6)Datacontext 4)DBML**
Down	**1)EntityClass 2) Association 3)Sqlmetal**

Unit-3

Across	**1)Skip 3)Quantification 4)Take 5)Asqueryable**
Down	**1)Ienumerator**

Unit-4

Across	**2)Xlinq 3) NodesBeforeself 4)Xnode**
Down	**1)Addmethod 2) XElement**

Unit-5

Across	**3)typeddatasets 4)Table 5)Deferred**
Down	**1)Gethashcode 2)Xsd**

Team Members

Srikanth Nivarthi
47275

Sreenivas Ram
66223

Seshu Babu Barma
56150

Radhika Shashidhar jaji
75217

Ashok Sharma
62751

Veerendra Kumar Ankem
77964

Teena Arora
74572

Contributors

Shalini Patil
46183

Aswin Raj Epari
72231