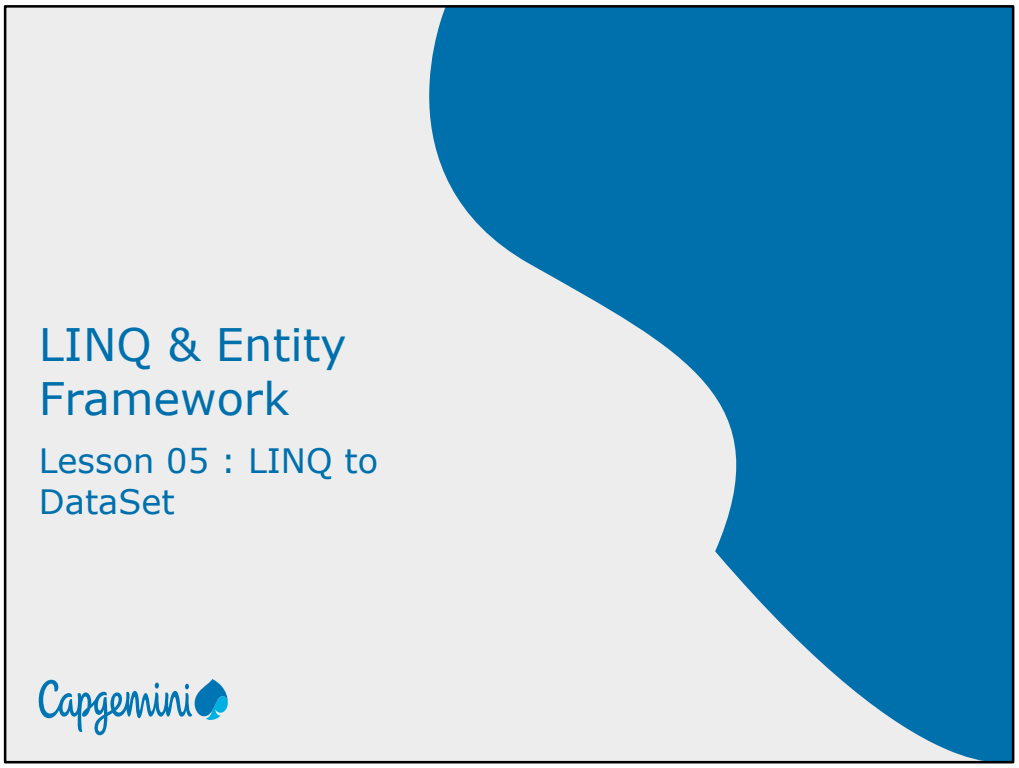
LINQ & Entity Framework
Lesson 05 : LINQ to DataSet
Capgemini

Lesson Objectives

- In this lesson we will cover the following
  - Understanding LINQ to DataSet
  - Implementing LINQ to DataSet

Understanding LINQ to DataSet

- .New LINQ Provider – LINQ to DataSet
  - .NET 3.5 comes with many LINQ providers out-of-the-box and one of them is LINQ to DataSets
- Performing LINQ Queries on DataSet & DataTable
  - LINQ to Dataset means performing a LINQ query operations on Dataset or DataTable
- Used to Perform LINQ Operations on ADO.NET Disconnected Architecture
- LINQ to DataSet is also know as LINQ to DataTable
  - Writing LINQ queries on records available in DataSet's DataTable object.

.NET Framework 3.5 comes with many LINQ providers out-of-the-box and one of them is LINQ to DataSets. Even though ADO.NET provides mechanisms in the form of methods and properties, LINQ to DataSets provides a easy and flexible option that resembles standard SQL in terms of language syntax to query and filter data in DataTable objects

LINQ to Dataset means performing a LINQ query operations on Dataset. Generally dataset is the most widely used component in ADO.NET because its built with disconnected architecture but its having limited querying capabilities. The LINQ to Dataset provides a facility to write richer queries on dataset based on our requirements.

Implementing LINQ to DataSet

- Call AsEnumerable() on DataTable to Access LINQ API
  - To support LINQ on DataTable, we have to call AsEnumerable() method on the DataTable, which returns an object implementing IEnumerable<T> interface.
- Write LINQ Query on IEnumerable<DataRow> using Field<T>()
  - Calling Field<T>() on DataRow element provides strongly-typed access to each of the column values in specified row.
- Can use CopyToDataTable() to convert LINQ result into DataTable
  - To convert the IEnumerable<DataRow> back to the DataTable we can use CopyToDataTable() method.

A LINQ query operation consists of three actions: obtain the data source or sources, create the query, and execute the query.

Data sources that implement the IEnumerable<T> generic interface can be queried through LINQ. Calling AsEnumerable on a DataTable returns an object which implements the generic IEnumerable<T> interface, which serves as the data source for LINQ to DataSet queries.

In the query, you specify exactly the information that you want to retrieve from the data source. A query can also specify how that information should be sorted, grouped, and shaped before it is returned. If the query is designed to return a sequence of values, the query variable itself must be a enumerable type. This query variable takes no action and returns no data; it only stores the query information. After you create a query you must execute that query to retrieve any data.

In a query that returns a sequence of values, the query variable itself never holds the query results and only stores the query commands. Execution of the query is deferred until the query variable is iterated over in a foreach or For Each loop. This is called *deferred execution*; that is, query execution occurs some time after the query is constructed. This means that you can execute a query as often as you want to. This is useful when, for example, you have a database that is being updated by other applications. In your application, you can create a query to retrieve the latest information and repeatedly execute the query, returning the updated information every time.

LINQ to DataSet – Sample Code

```
//Obtaining the data source on which to write LINQ
var empDataSource = GetDataSet().Tables["Emp"].AsEnumerable();
//LINQ query to get employees having salary greater than 38000
var query = from r in empDataSource
            where r.Field<decimal>("Salary") > 38000
            orderby r.Field<decimal>("Salary")
            select r;
//executing the query to retrive the LINQ result
foreach (var row in query)
{
    string str = $"EmpId: {row["EmpId"]}, Name: {row["EmpName"]},
                 Salary: {row["Salary"]}";
    Console.WriteLine(str);
}
//Loading the query results into DataTable object.
DataTable table = query.CopyToDataTable();
```

In the above code snippet, we will first obtain top level data source which is DataSet object. To get DataSet we are calling GetDataSet() user defined method which fills the dataset by retrieving data from database.
Then we will call AsEnumerable on a DataTable object from DataSet to get IEnumerable<DataRow> so then we can write any LINQ functionality on that.
After writing LINQ query you will get result in IEnumerable<DataRow> variable, which you can iterate through foreach or else you can also convert it into DataTable object by calling CopyToDataTable() method on IEnumerable<DataRow>.

Some more example for LINQ to DataSet are given below.

```
//Sample-#2
var query = from recEmp in empDataSource
        where recEmp.Field<DateTime>("DOJ").Year == 2003
        select new
        {
            EmpId = recEmp.Field<int>("EmpId"),
            EmpName = recEmp.Field<string>("EmpName"),
            Gender = recEmp.Field<string>("Gender"),
            DOJ = recEmp.Field<DateTime>("DOJ")
        };
foreach (var emp in query)
{
    var strEmp = $"EmpId: {emp.EmpId}, Name: {emp.EmpName}, Gender:
{emp.Gender}, DOJ: {emp.DOJ}";
    Console.WriteLine(strEmp);
```

```
}

//Sample-#3
var query = from recEmp in empDataSource
            where recEmp.Field<string>("Gender") == "Male"
            orderby recEmp.Field<DateTime>("DOJ")
            select new
            {
               EmpId = recEmp.Field<int>("EmpId"),
               EmpName = recEmp.Field<string>("EmpName"),
               Gender = recEmp.Field<string>("Gender"),
               DOJ = recEmp.Field<DateTime>("DOJ")
            };
foreach (var emp in query)
{
   var strEmp = $"EmpId: {emp.EmpId}, Name: {emp.EmpName}, Gender: {emp.Gender}, DOJ:
{emp.DOJ}";
   Console.WriteLine(strEmp);
}
```

Demo

- Demo of Implementing standard operator in LINQ

Summary

- In this lesson we have learnt the following topic
  - What is LINQ to DataSet
  - How to write LINQ Query on DataTable object