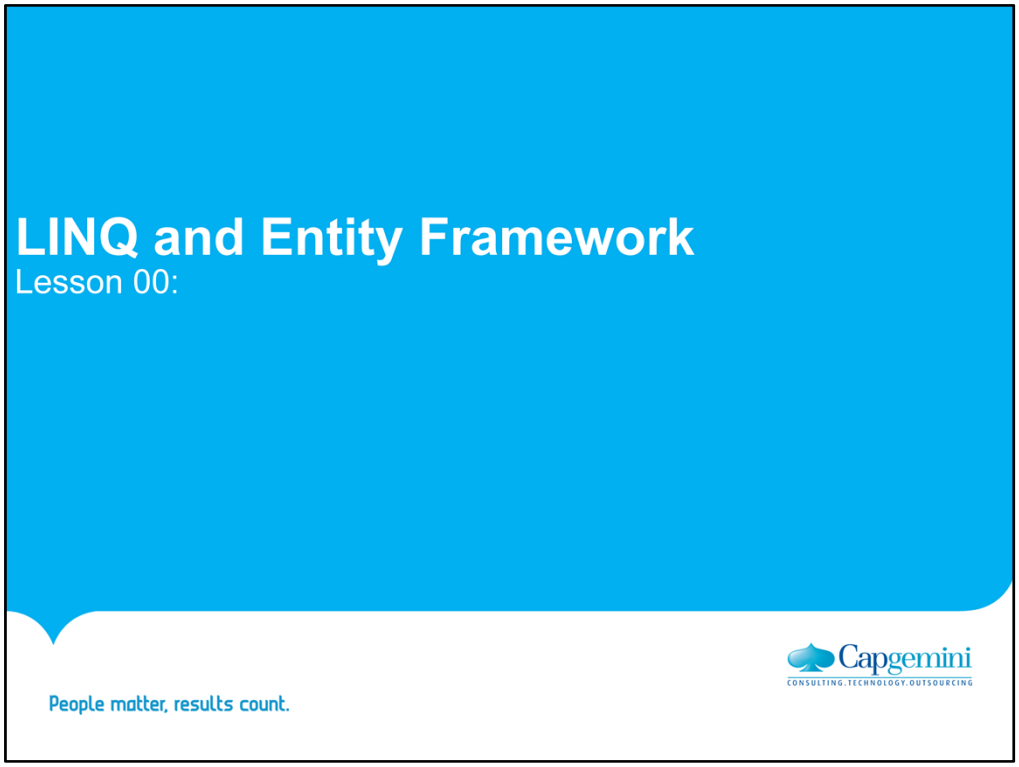


Document History

Date	Course Version No.	Software Version No.	Developer / SME	Change Record Remarks
		Microsoft Visual Studio 2013	Vijay Vishwakarma	New

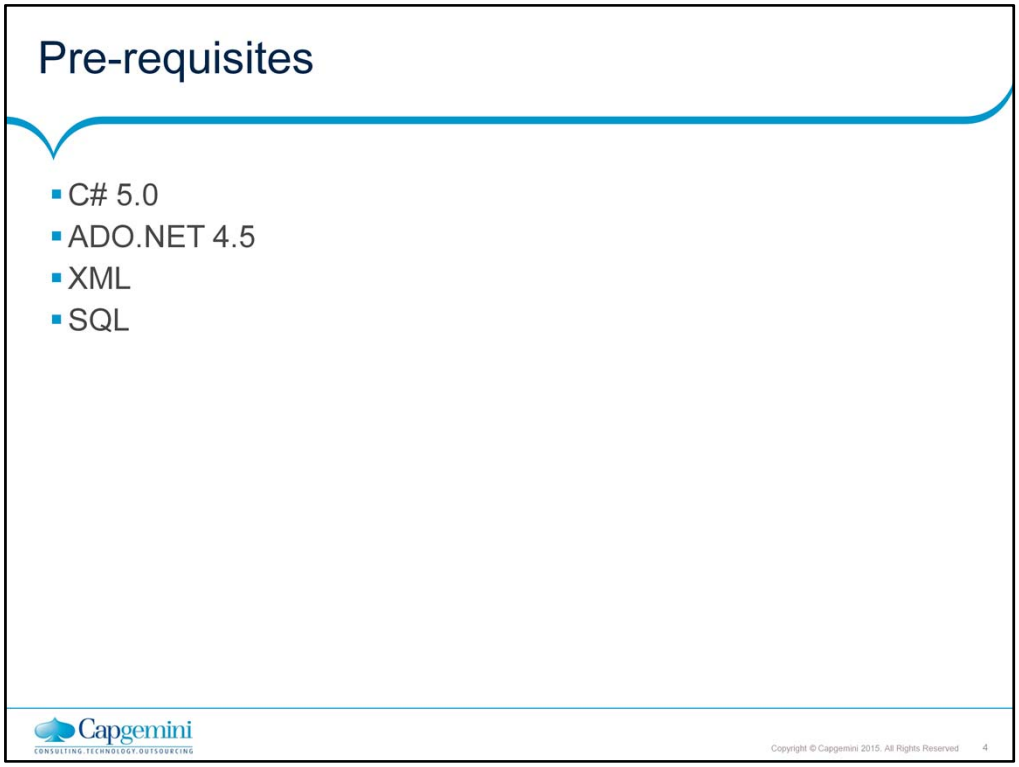
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Course Goals and Non Goals

- Course Goals
  - To learn LINQ and Entity Framework as Data Access Technology
  - To learn use of LINQ in Entity Framework
- Course Non Goals
  - Learning ADO.NET

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3


Pre-requisites
C# 5.0
ADO.NET 4.5
XML
SQL
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
4

Intended Audience

- Software Developers, Engineers etc

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

Day Wise Schedule

- Day 1
  - Lesson 1: Introduction to LINQ and LINQ to Objects
  - Lesson 2: Introduction to Entity Framework 6.0
- Day 2
  - Lesson 3:EDM Tool Enhancement
  - Lesson 4: EF Runtime Features
  - Lesson 5: Overview of LINQ to Entities

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

Table of Contents

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

References

- Student Guide
  - PPT along with notes
- Lab Book
  - Lab Book consists of Walkthroughs to guide the participants throughout and <<To Do>> Exercises to test their knowledge and skills
- References
  - http://msdn.microsoft.com
  - http://www.asp.net/entity-framework
  - https://msdn.microsoft.com/en-in/data/ef.aspx
  - Entity Framework 6 Recepies by Apress publication

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Next Step Courses (if applicable)

- Asp.Net
- Asp.Net MVC
- WEB API

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

LINQ and Entity Framework

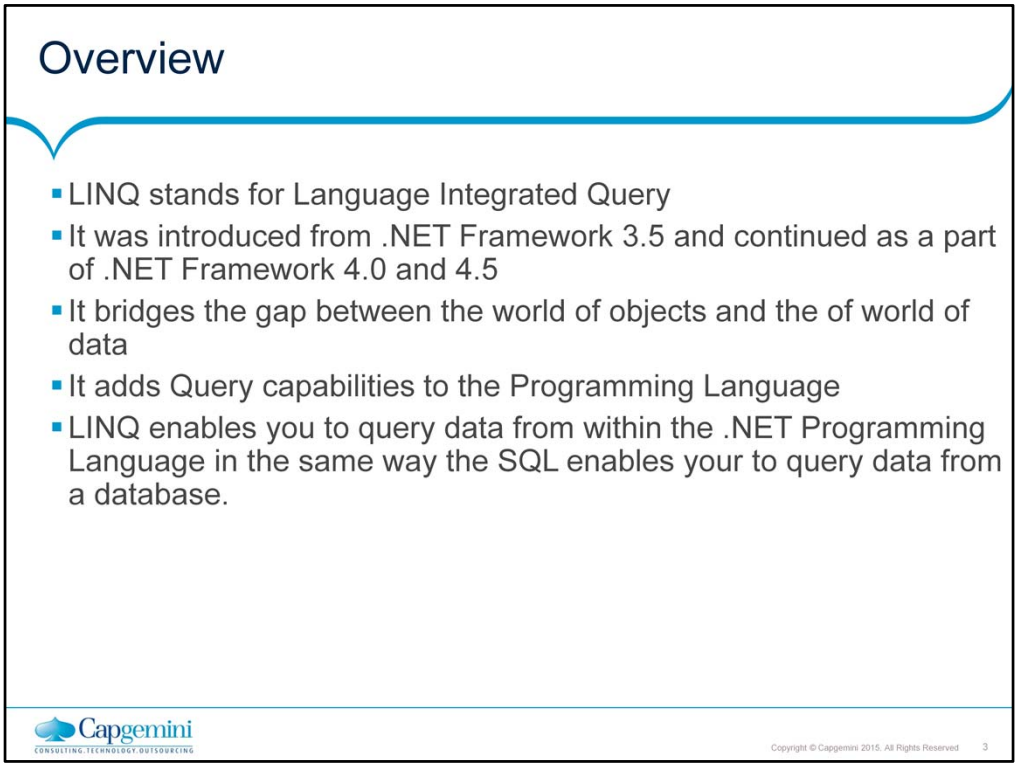
Lesson 01: Introduction to LINQ and LINQ to Objects

Lesson Objectives

- In this lesson we will cover the following
  - Overview of LINQ
  - LINQ Architecture and Components
  - Basic Query Operations with LINQ

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



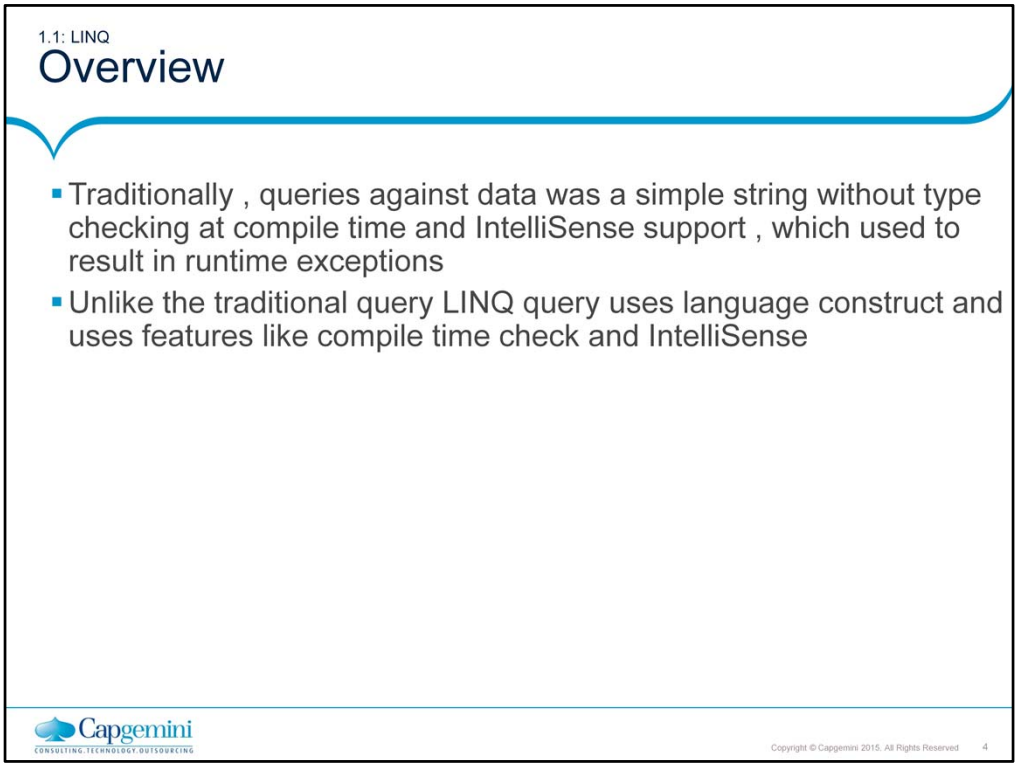
Traditionally ,Developer use to write down code for accessing database object ADO.NET where they use to pass SQL query as string to the database which does not have a compile time due which if the query if formed incorrect it use to result in Runtime exception and abruptly putting a stop to Program execution flow .
To over come this problem LINQ was introduced as a part of .NET framework 3.5 which allowed the developer to write query to handle data using programming language like C#.
With the introduction of **L**anguage **In**tegrated **Q**uery (LINQ), the developers now can:

- deal with data by using a consistent programmatic approach.
- perform data access with new data design surfaces.

LINQ aims to reduce the complexity for developers. It helps boost their productivity through a set of extensions to the C# and Visual Basic programming languages, as well as the Microsoft .NET Framework, which provides integrated querying for objects, databases, and XML data.
Using LINQ, developers are able to write queries natively in C# or Visual Basic without using specialized languages, such as Structured Query Language (SQL) and Xpath.
With Visual Studio , you can work with data in the way you want, namely:

- You can create entire collections of objects from a database backend, if required.
- You can interact with data as rows or columns – whatever makes sense to your application.

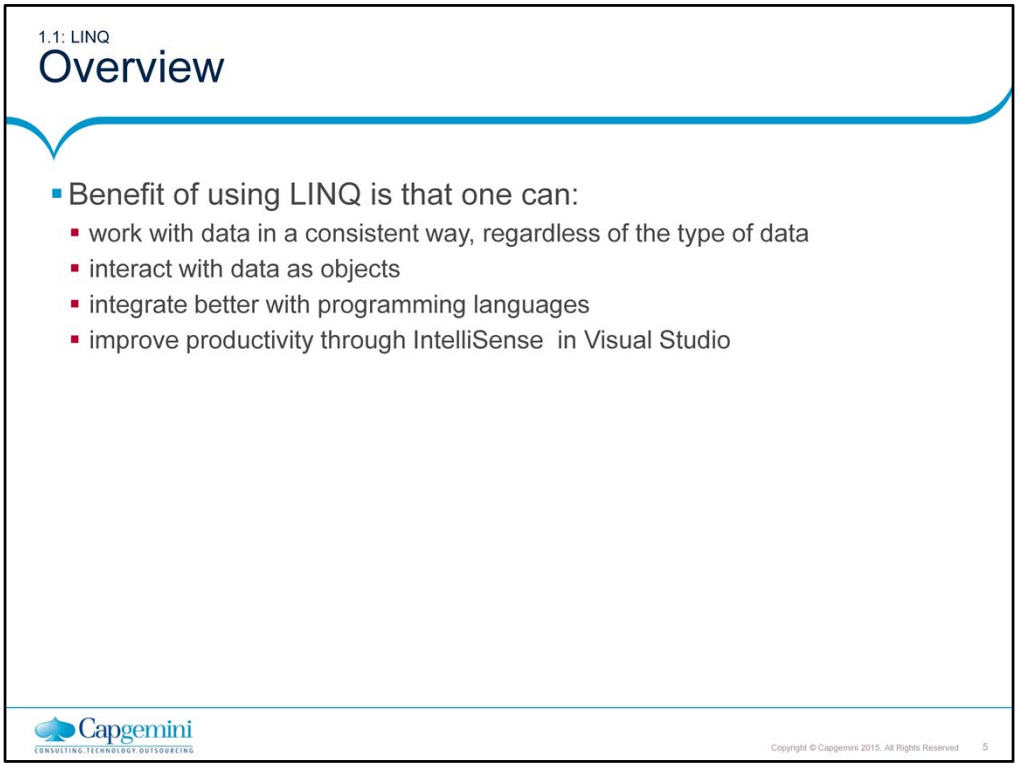
Language Integrated Query or “LINQ” dramatically changes the way we work with and program against data. By creating a common querying language in LINQ, the developer is free
to focus on things that matter most. LINQ provides the ease of use that you have come to expect with Visual Studio offering both “IntelliSense” and “Autocompletion” right in the IDE.

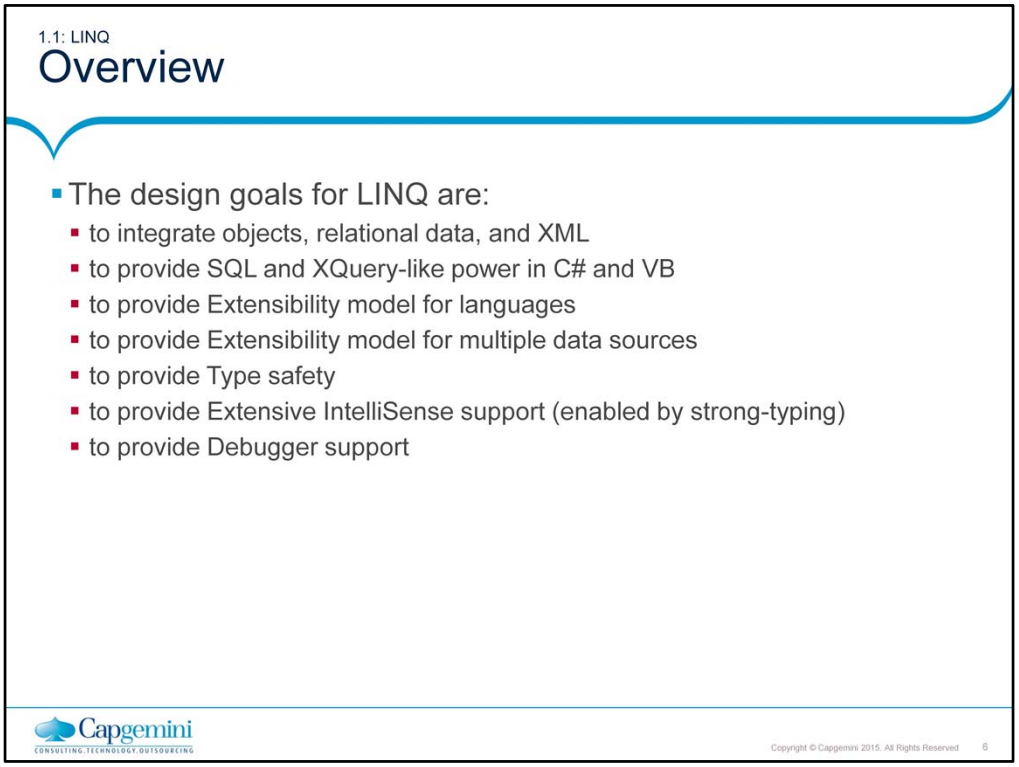


LINQ provides native querying syntax in C# and VB.Net. This frees the developer from having to master independent data programmability technologies (for example: Xpath, Xquery, T/SQL). Instead, LINQ offers the developer a consistent way to query data.

The best part is that the LINQ code that you write is consistent. This is irrespective of whether the data store is one of the following:

- a SQL Server
- a data store contained in a ADO.NET DataSet
- an XML document
- an EDM that you create, or
- an object that you create in memory

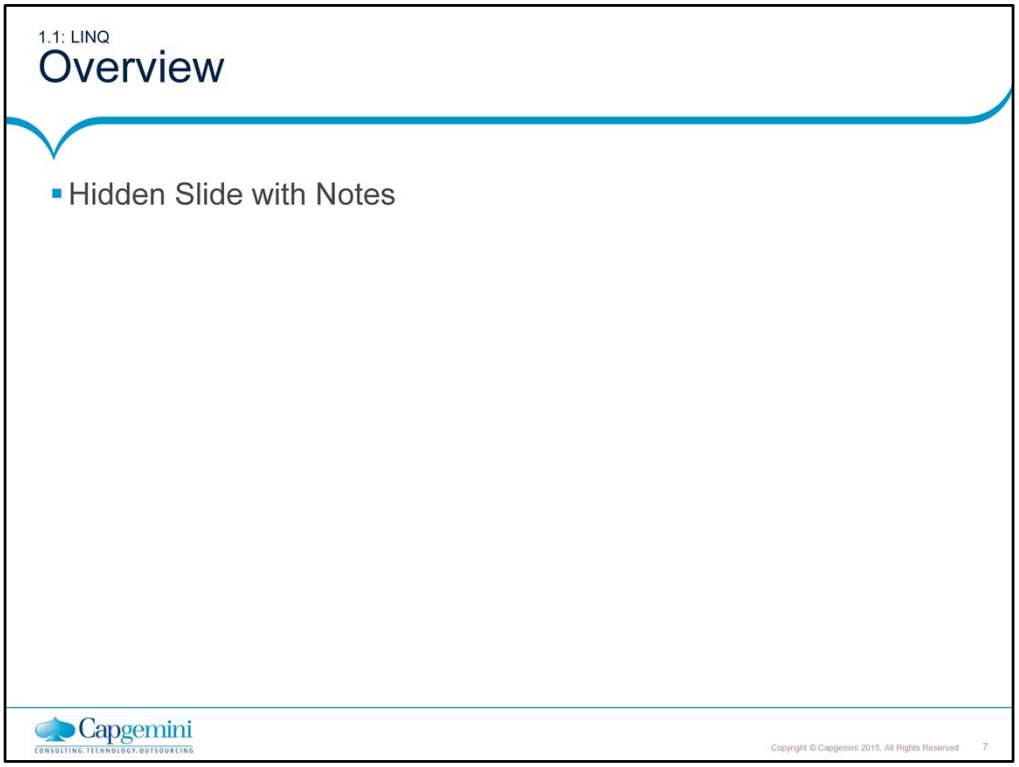

1.1: LINQ
Overview
Benefit of using LINQ is that one can:
work with data in a consistent way, regardless of the type of data
interact with data as objects
integrate better with programming languages
improve productivity through IntelliSense in Visual Studio
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 5



LINQ- Design Goals:

Design goals for LINQ are:

- **To integrate objects, relational data, and XML**
  - To provide unified query syntax across data sources to avoid different languages for different data sources.
  - To provide Single model for crunching all types of data regardless of source or in-memory representation.
- **To provide SQL and XQuery-like power in C# and VB**
  - To integrate querying abilities right into the programming languages.
- **To provide Extensibility model for languages**
  - To enable implementation for other programming languages.
- **To provide Extensibility model for multiple data sources**
  - To be able to access other data sources than relational databases or XML documents.
  - To allow other frameworks to implement LINQ for their own needs.
- **To provide Type safety**
  - To provide “compile-time type checking” to avoid problems that were previously discovered only at run-time.
  - To enable the compiler to catch errors in your queries.

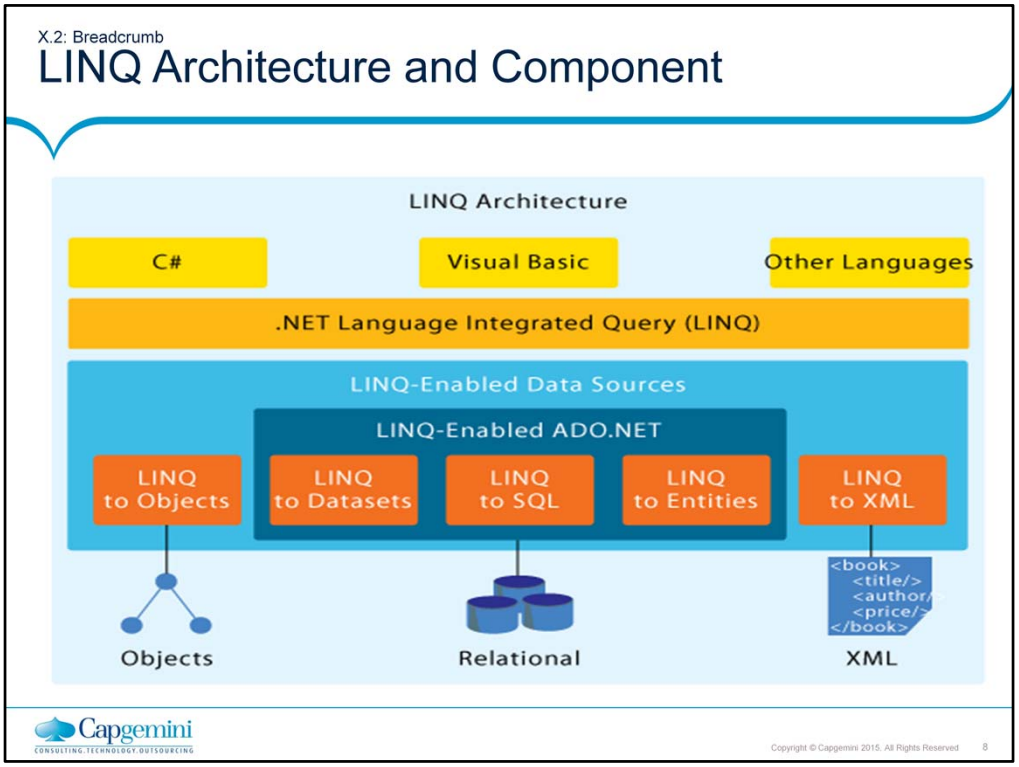


Design Goals of LINQ (contd.):

- **To provide extensive IntelliSense support (enabled by strong-typing)**
  - To assist the developers when writing queries to improve productivity
  - To help the developers get up to speed with the new syntax. (The editor guides when writing queries.)
- **To provide Debugger support:**
  - To provide Debugger support that allows the developers to debug LINQ queries in a step by step fashion along with rich debugging information.

- The number one LINQ feature is the ability to deal with several data types and sources. LINQ ships with implementations that support querying against regular object collections, databases, entities, and XML sources. Since LINQ supports "rich extensibility", the developers can easily integrate LINQ with other data sources and providers, as well.

- Another essential feature of LINQ is that it is "strongly-typed". This means that:
  - You get compile-time checking for all queries, unlike contemporary SQL statements, where you typically find out only at run-time in case something is wrong. This implies that you will be able to check during development that your code is correct. The direct benefit is a reduction in the number of problems discovered late in production. Usually, bugs arise due to human factors. Strongly-typed queries allow early detection of typos, and mistakes done by the developer in charge of the keyboard.
  - You get IntelliSense within Visual Studio when writing LINQ queries. This makes typing faster. It also makes working easier - against both simple and complex collection and data source object models.



LINQ Architecture and Components:-

LINQ is currently integrated directly in the native syntax for C# 5.0 and VB 10 which are part of .Net Framework 4.5 and Visual Studio

The above diagram show the architecture of the LINQ and all the components of LINQ which are as follows

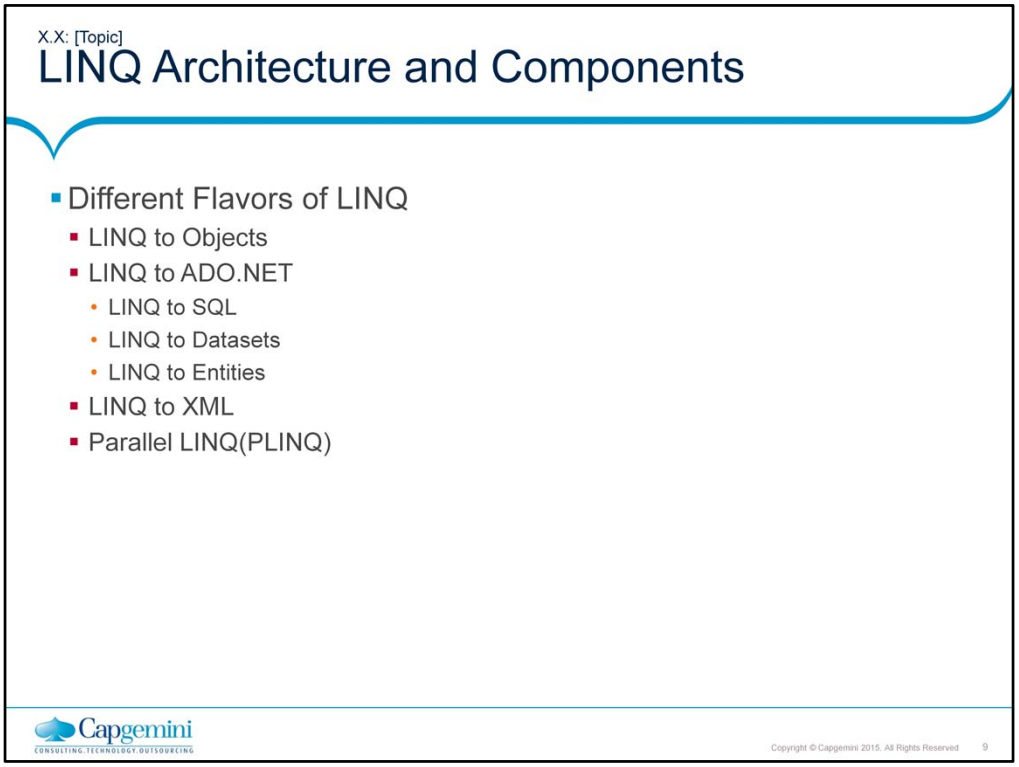
The top most layer consist of language extension which show all the languages which can use LINQ including C# , VB and other langugage which are the part of .Net Framework

The second layer consist of .NET LINQ which

The third layer consist of LINQ Enabled Data source which specifies different data source on which LINQ can be used . LINQ can be use with Objects, SQL, Dataset and XML.

LINQ comes in different flavors based on data source being used as follows

i) LINQ to Objects
ii) LINQ to ADO.Net which includes LINQ to Datasets, LINQ to SQL, LINQ to Entities
iii) LINQ to XML



LINQ to Objects:

- SQL-like queries for any .NET collection (or anything that implements Ienumerable)
- The LINQ to Objects API supports queries over any .NET collection, such as arrays and generic lists.
- This API is defined in the System.Linq namespaces inside System.Core.dll
- LINQ to objects is enabled by including the System.Linq namespace.
- Manipulating collections of objects, which can be related to each other to form a hierary or a graph. From a certain point of view, LINQ to Objects is the default implementation used by a LINQ query.

LINQ to ADO.NET:

- Query enabled data access framework
- Allows to perform query operation on Sql, DataSet and Entities
- API is defined in System.Data.Linq namespace

LINQ to XML:

- Query enabled, smaller, faster XML DOM
- API is defined in System.XML.Linq namespace

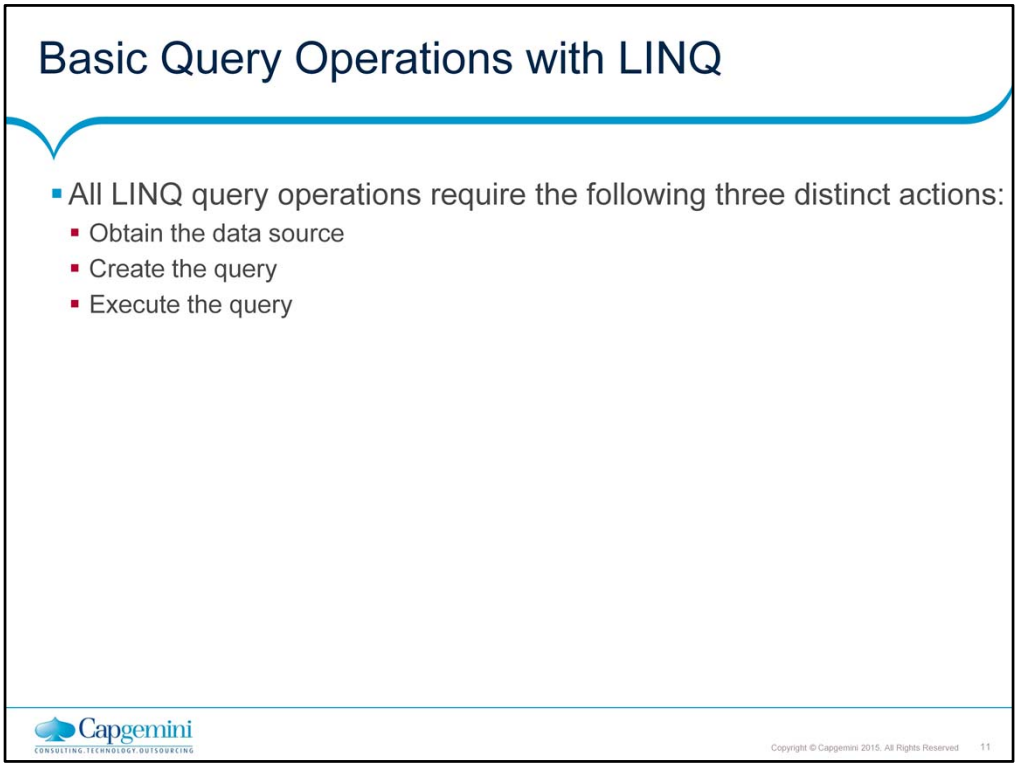
- One important thing to remember is that querying syntax use in LINQ remains consistent regardless of the type of data you are working with

Basic Query Operations with LINQ

- LINQ Queries are written using the LINQ declarative query syntax.
- These queries uses a set of query keywords built into the .NET Framework that allows the developer to write SQL like commands in programming language.
- Commonly used Keywords are
  - from / in
  - where
  - orderby
  - select
  - groupby

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10



Basic Operations in LINQ:
A query is an expression that retrieves data from a data source. Queries are usually expressed in a specialized query language. Different languages have been developed over time for the various types of data sources.
For example: SQL for relational databases and XQuery for XML.

Therefore developers have had to learn a new query language for each type of data source or data format that they must support. LINQ simplifies this situation. It offers a consistent model for working with data across various kinds of data sources and formats. In a LINQ query, you are always working with objects. You use the same basic coding patterns to query and transform data in XML documents, SQL databases, ADO.NET Datasets, .NET collections, and any other format for which a LINQ provider is available.

All LINQ query operations consist of three distinct actions:
1. Obtain the data source.
2. Create the query.
3. Execute the query.

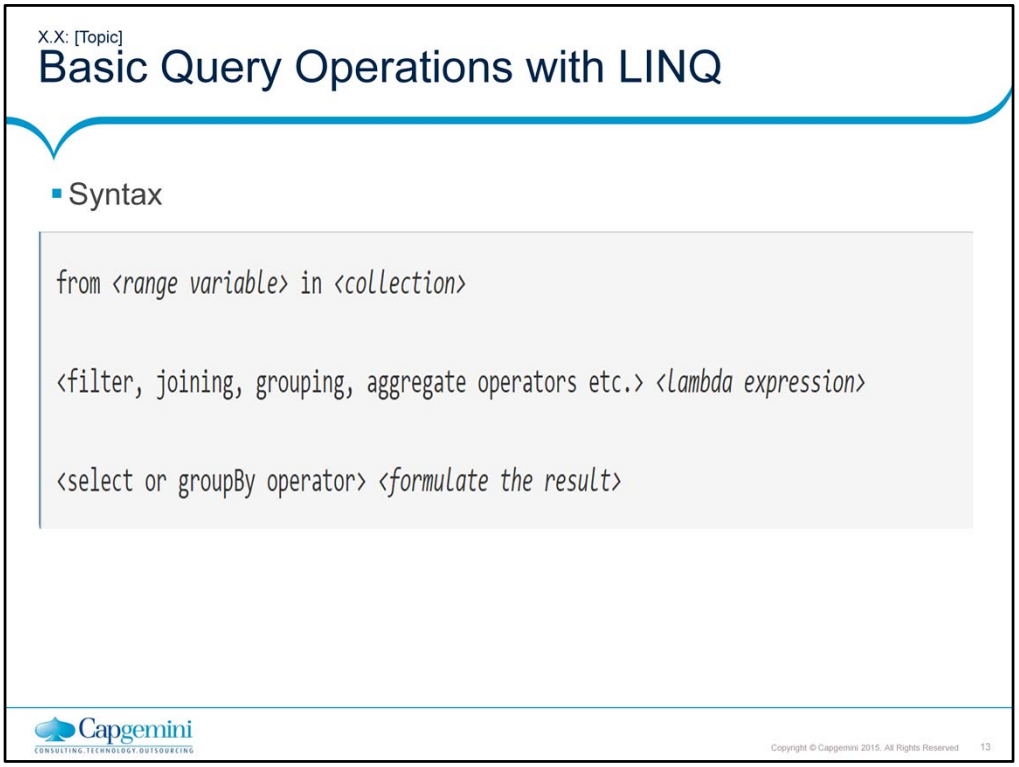
Example

```
class IntrotoLINQ
{
   static void Main()
   {
            //1.Data Source
            int [] numbers = new int[7]{0,1,2,3,4,5,6};

            //2.Query Creation
            var numQuery= from num in numbers
                                    where (num%2)==0
                                    select num;
            //3.Query Execution
            foreach(int num in numQuery)
            {Console.Write("{0,1}",num);}
   }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

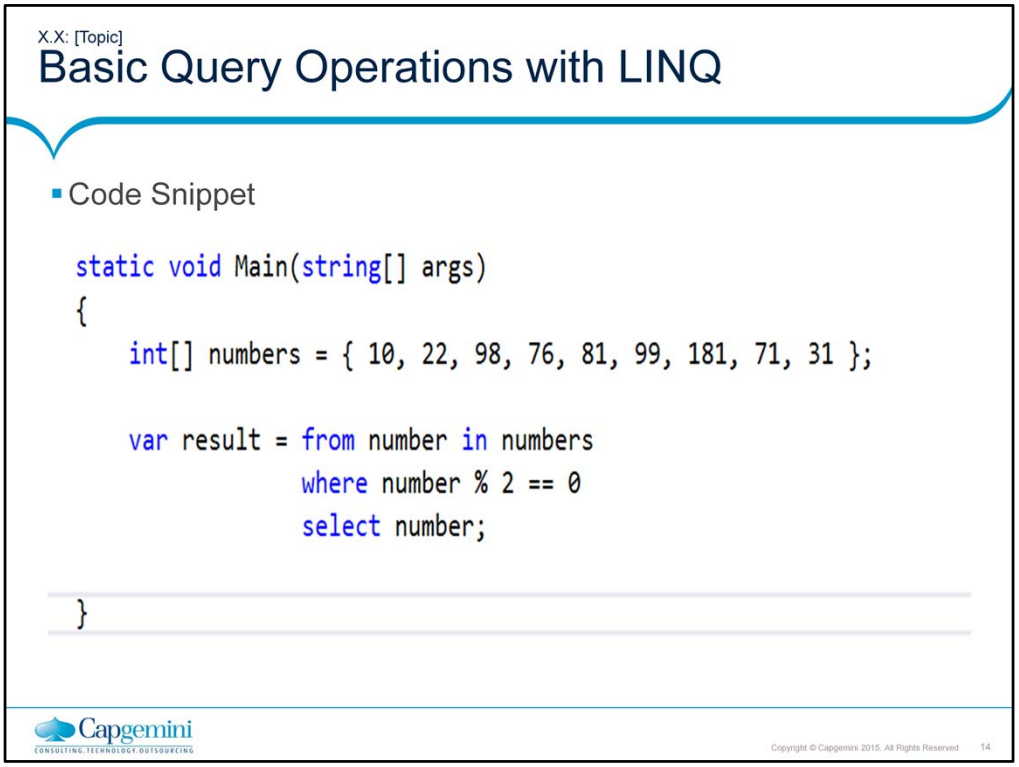
 12



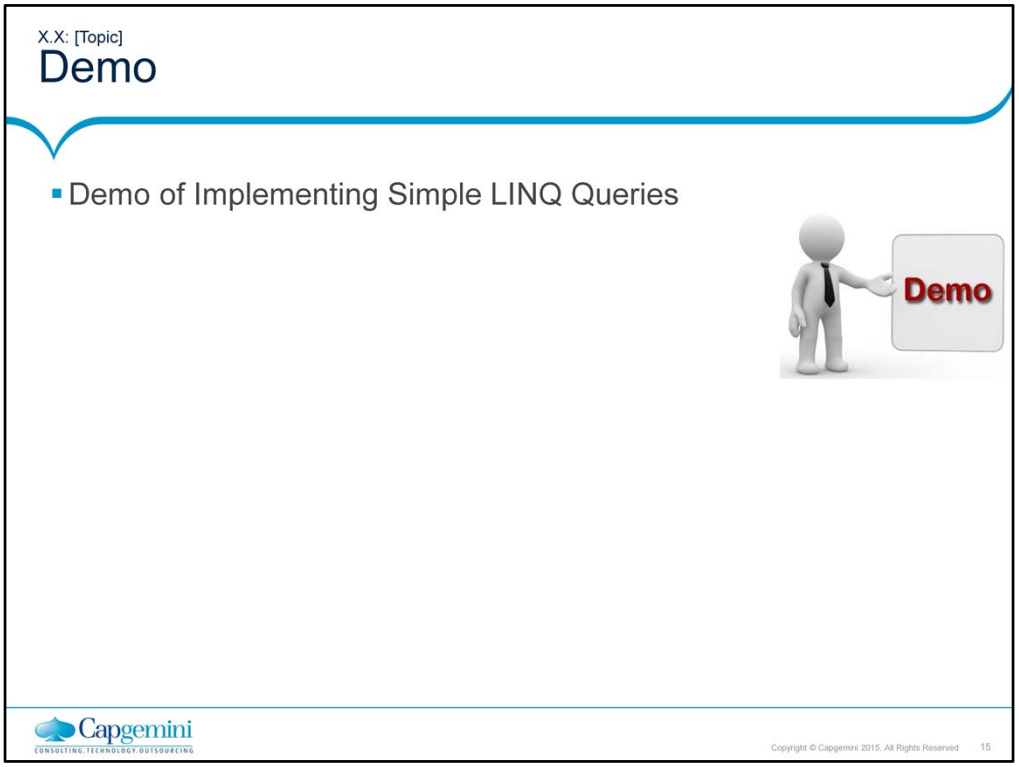
In the above diagram we can see the syntax of a Basic LINQ Query

Every LINQ query begins with a from clause ends with a select query or group. In between can be use filtering , joining , grouping etc on the data.

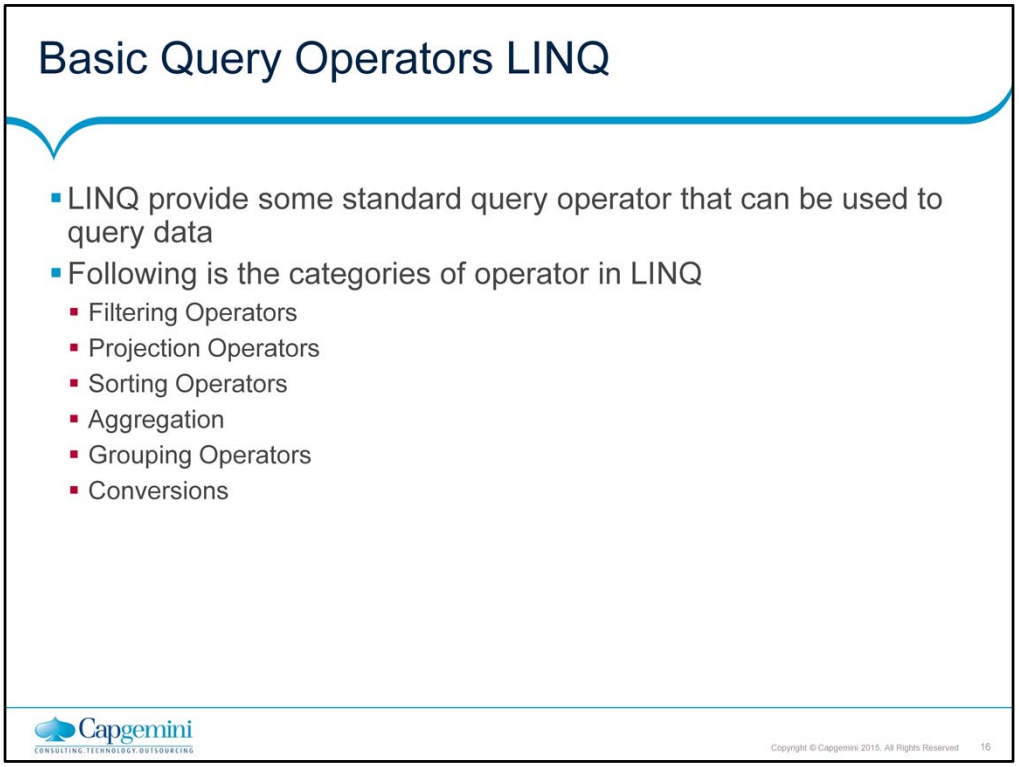
Query looks similar to SQL and will provide intellisense support.



Add the notes here.
The Code Snippet in the Slide returns even number from the Integer array and stores them in variable name result



Add the notes here.
Demo of Simple LINQ Queries.



Basic Query Opertors

Filtering Operators

i) Where :- Filter value based on a predicate function

Projection Operators

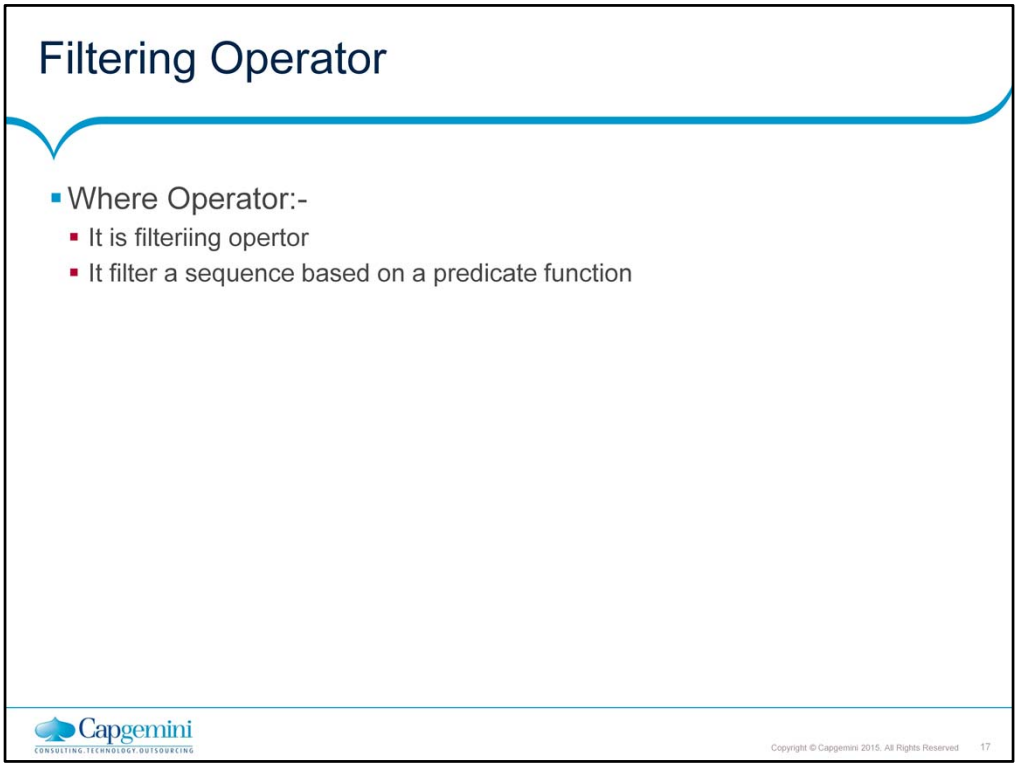
i) Select :- The operator projects value on basis of a transform function

Sorting Operators

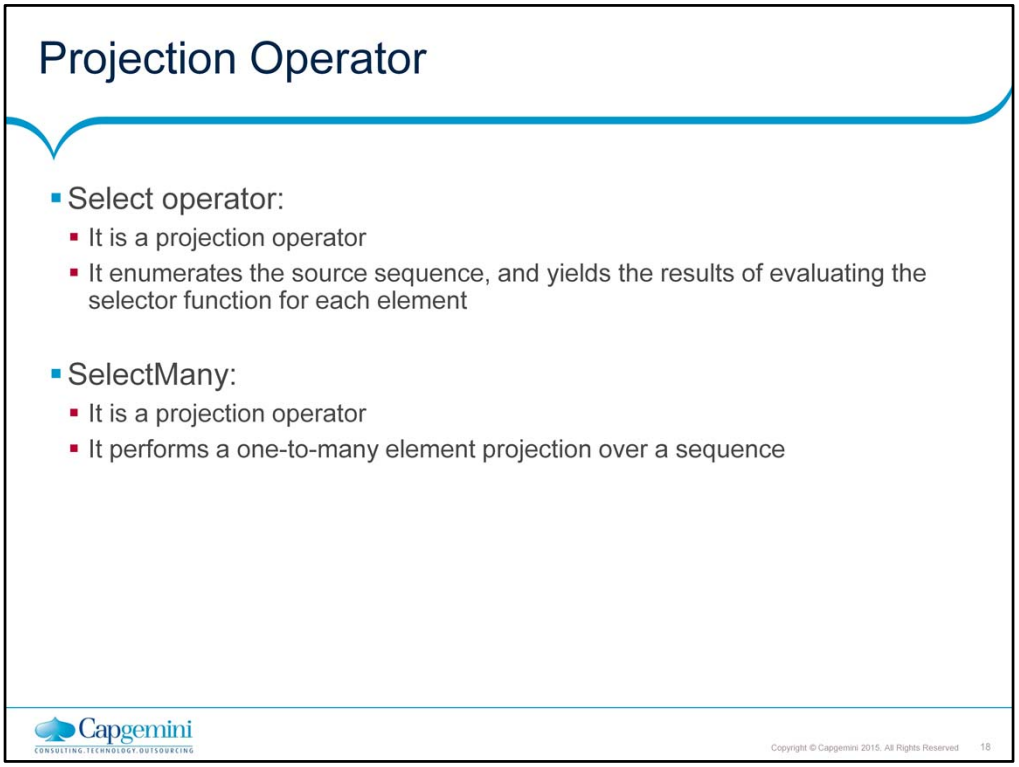
i) OrderBy :- The operator sort values in an ascending order
ii) OrderByDescending :- The operator sort values in a descending order
iii) ThenBy:- Executes a secondary sorting in an ascending order
iv) ThenByDescending :- Executes a secondary sorting in a descending order
v) Reverse :- Performs a reversal of the order of the elements in a collection

Aggregation

i) Average :- Average value of collection of value is calculated
ii) Count :- Counts the number of entries
iii) Max :- Find out the maximum value within a collection
iv) Min :- Find out the Minimum value within a collection
v) Sum :- Find out the sum of value within a collection



The Where Operator:
Suppose that you have to list the names and cities of customers from Italy. To filter a set of items, you can use the Where operator, which is also called a “restriction operator” because it restricts a set of items.



Projection Operators:

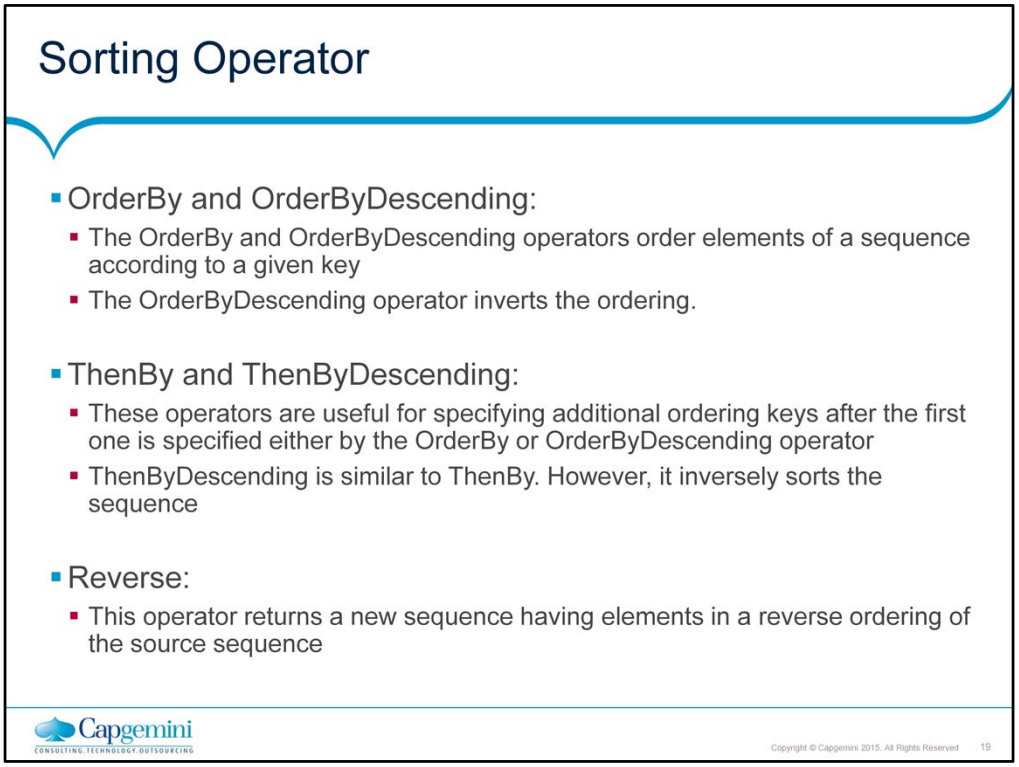
Select:

- The Select operator performs a projection over a sequence.
- When the object returned by Select is enumerated, it enumerates the source sequence. Subsequently, it yields the results of evaluating the select or function for each element.
  - The first argument of the selector function represents the element to process.
  - The second argument, if present, represents the zero-based index of the element within the source sequence.

SelectMany:

The SelectMany operator performs a projection over a sequence. This operator is similar to Select because it allows us to define the elements that have to be picked up from a sequence. The difference is in the return type.

With the IEnumerable<S> type returned by the selector parameter of SelectMany, it is possible to concatenate many projection operations together. This can be done either on different sequences or starting from the result of a previous query.



Sorting Operators:

OrderBy and OrderByDescending:

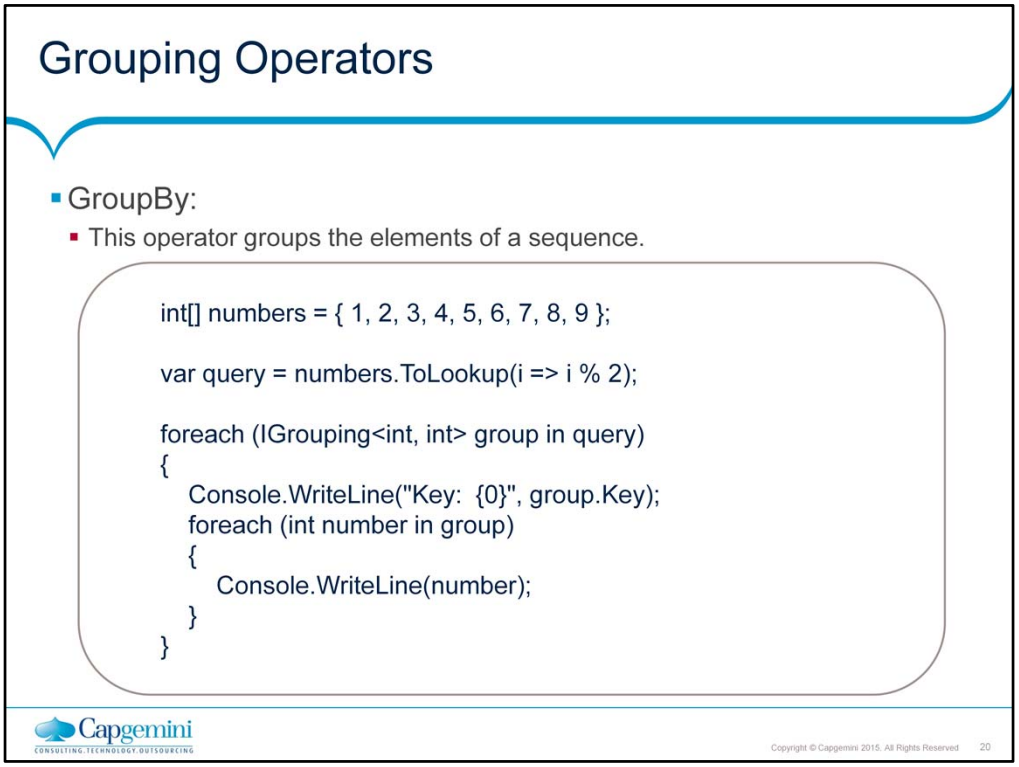
- Similar to the ORDER BY and ORDER BY DESC in SQL, the OrderBy and OrderByDescending operators order elements of a sequence according to a given key. The OrderByDescending operator inverts the ordering.

ThenBy and ThenByDescending:

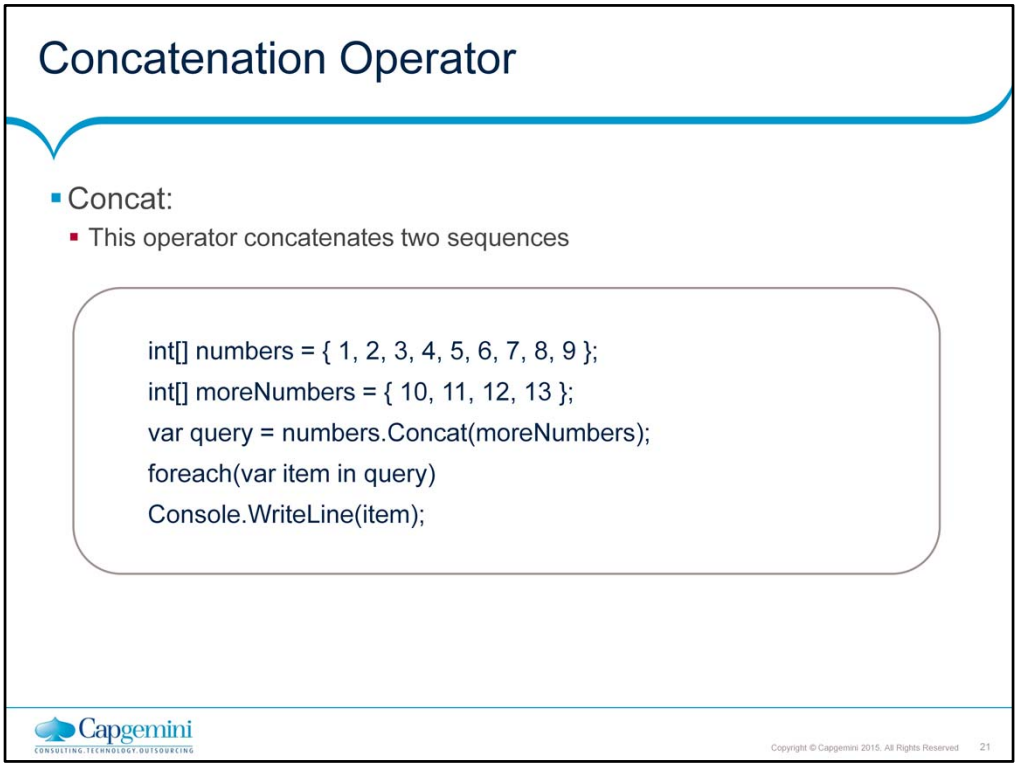
- OrderBy allows us to specify only one ordering key. Hence we have to use either ThenBy or ThenByDescending to concatenate ordering-key values.

Reverse:

- This method simply returns a new sequence with elements in a reverse ordering of the source sequence.

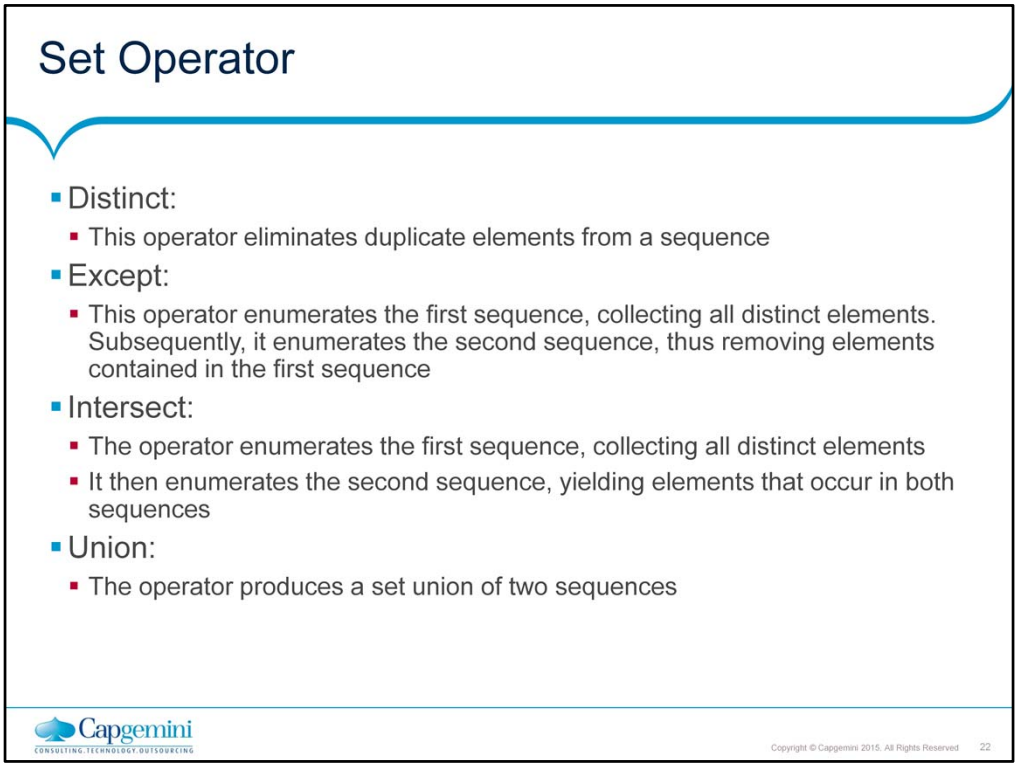


Grouping Operators:
Similar to the GROUP BY clause of SQL, the GroupBy operator groups elements of a sequence based on a given selector function.



Concatenation Operators:

- This operator concatenates two sequences.
- The resulting IEnumerable<T> type is the concatenation of the first and second sequences specified as a parameter.



Set Operators:

Distinct:

- This operator is similar to the DISTINCT keyword used in SQL. It eliminates the duplicates from a sequence.
- When the code processes the query, it enumerates the element of the sequence, and stores into an IEnumerable<T> type. It is done for each element that has not been previously stored.
- The Distinct operator selects unique values from the sequence.

Except:

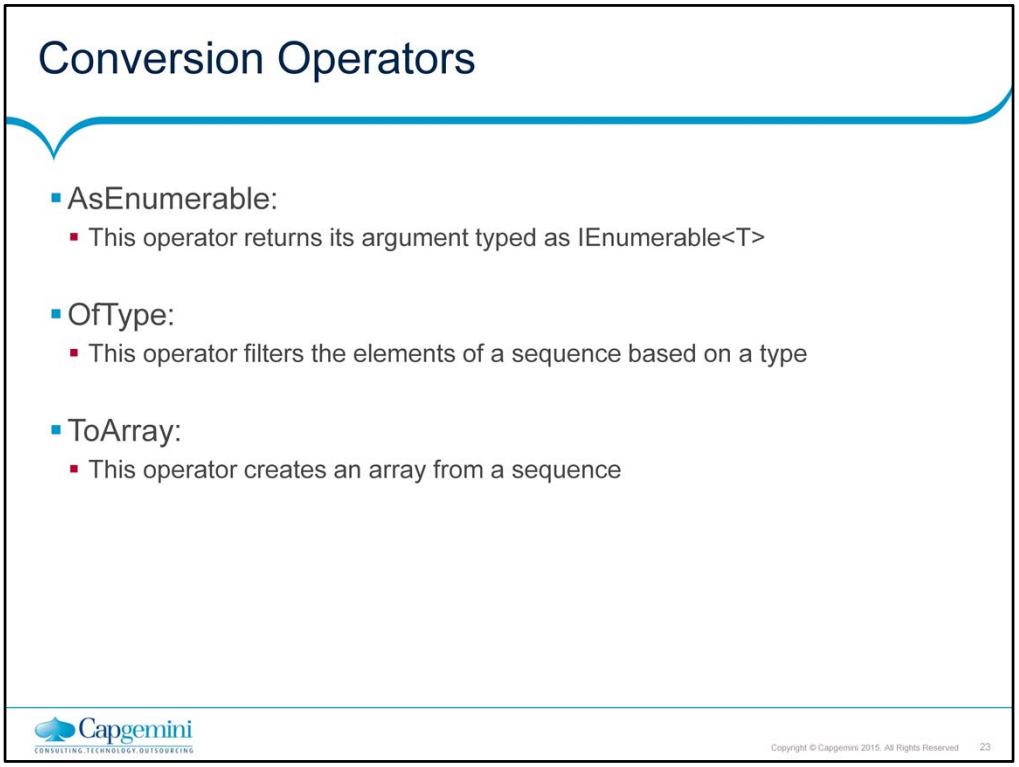
- This Operator Produces the set difference of two sequences by using the default equality comparer to compare values.

Intersect:

- This operator returns a sequence made by common elements of two different sequences.
- The Intersect Method is used to retrieve common elements in two sequences as shown below:

Union:

- This operator returns a new sequence formed by uniting the two different sequences.



Conversion Operators:

AsEnumerable
- This operator produces a new IEnumerable<T> type composed of only the element of the specified type

OfType:
- The OfType Searches for the specified type T in the Sequence

ToArray:
- This operator returns an array composed of the elements of the source sequence.

Aggregate Operators

- Aggregate:
  - The operator applies a function over a sequence

```
int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
var query = numbers.Aggregate((a, b) => a * b);
Console.WriteLine(query);
```

- Average:
  - The operator computes the average of a sequence of numeric values Count & LongCount
  - It counts the number of elements in a sequence

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

Aggregate Operator (Contd..)

- Max:
  - The operator finds the maximum value from a sequence of numeric values
- Min:
  - The operator finds the minimum value from a sequence of numeric values
- Sum:
  - The operator computes the sum of a sequence of numeric values

Element Operators

- First:
  - The operator returns the first element of a sequence
- FirstOrDefault:
  - The operator returns the first element of a sequence, or a default value if no element is found
- Last:
  - The operator returns the last element of a sequence
- LastOrDefault:
  - The operator returns the last element of a sequence, or a default value if no element is found

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

Element Operators (Contd..)

- Single:
  - The operator returns the single element of a sequence. An exception is thrown if the source sequence contains no match or more than one match
- SingleOrDefault:
  - The operator returns the single element of a sequence, or a default value if no element is found. The default value is for reference and nullable types

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Element Operators (Contd..)

- DefaultIfEmpty:
  - The operator supplies a default element for an empty sequence. It can be combined with a grouping join to produce a left outer join
- ElementAt:
  - It returns the element at a given index in a sequence
- ElementAtOrDefault:
  - The operator returns the element at a given index in a sequence, or a default value if the index is out of range

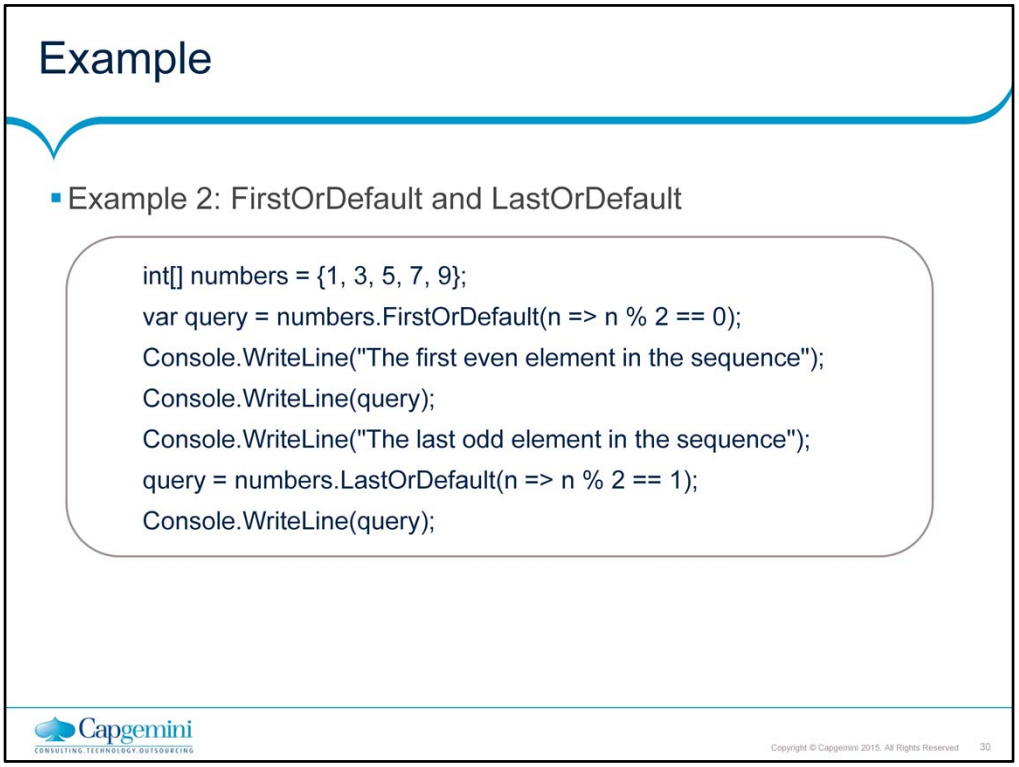
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Example

- Example 1:

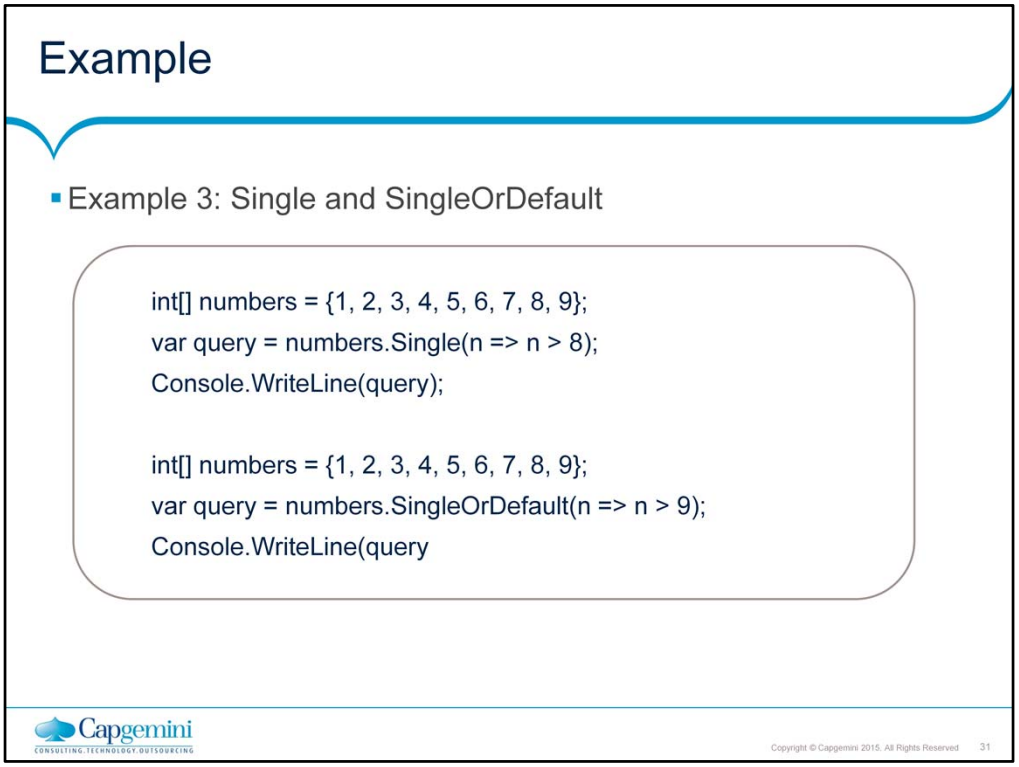
```
int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
var query = numbers.First();
Console.WriteLine("The first element in the sequence");
Console.WriteLine(query);
query = numbers.Last();
Console.WriteLine("The last element in the sequence");
Console.WriteLine(query);
Console.WriteLine("The first even element in the sequence");
query = numbers.First(n => n % 2 == 0);
Console.WriteLine(query);
Console.WriteLine("The last even element in the sequence");
query = numbers.Last(n => n % 2 == 0);
Console.WriteLine(query);
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



- Using the FirstOrDefault/LastOrDefault methods we obtain the same results. However, when we use those methods, and a predicate does not find an element satisfying the specified condition, a default value is returned (thereby avoiding retrieval of an exception).

- In the above example , Since no even numbers are in the sequence, FirstOrDefault returns the zero default value. On the other hand, the LastOrDefault operator looks for the last odd number in the sequence, and finds the number 9.



- Using the Single method, if no element satisfies the predicate condition, an exception is thrown.
- Using the SingleOrDefault method, when no element satisfies the predicate function, either a null or zero value is returned.
- The difference between the null and zero value depends on the source type: null for reference types (i.e., strings) and zero for value types (i.e., integers).

Example

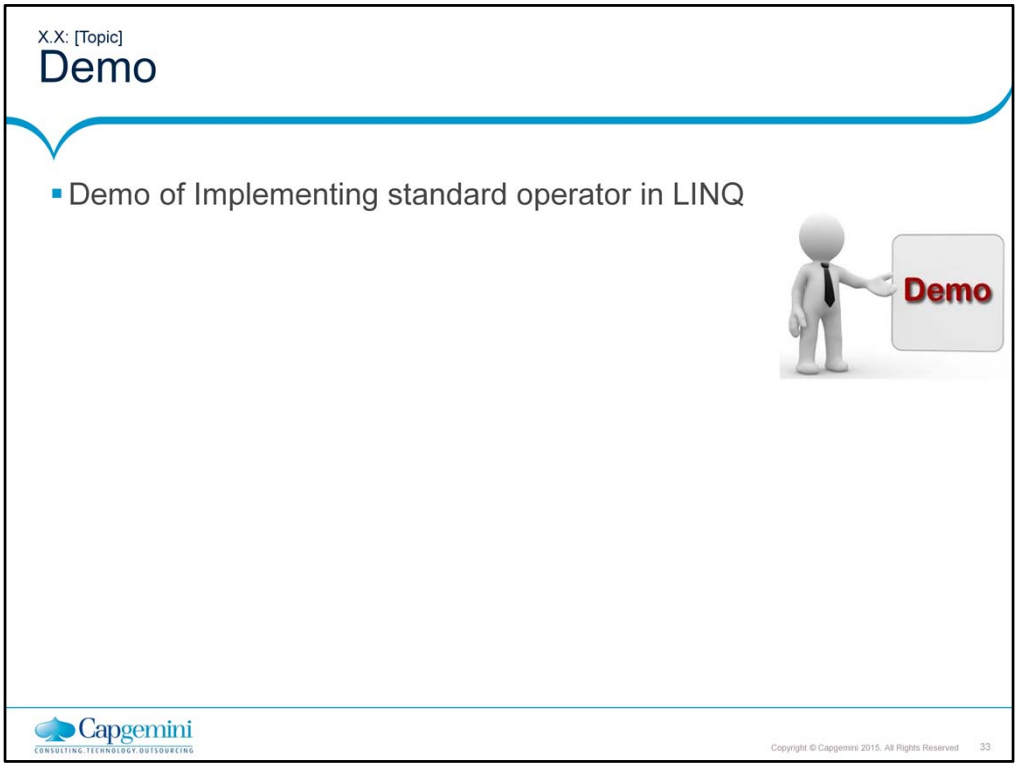
- Example 4: ElementAt and ElementAtOrDefault

```
int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9};
var query = numbers.ElementAt(4);
Console.WriteLine(query);

int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9};
var query = numbers.ElementAtOrDefault(9);
Console.WriteLine(query);
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 32



Add the notes here.
Demo of Simple LINQ Queries.

X.X: [Topic]

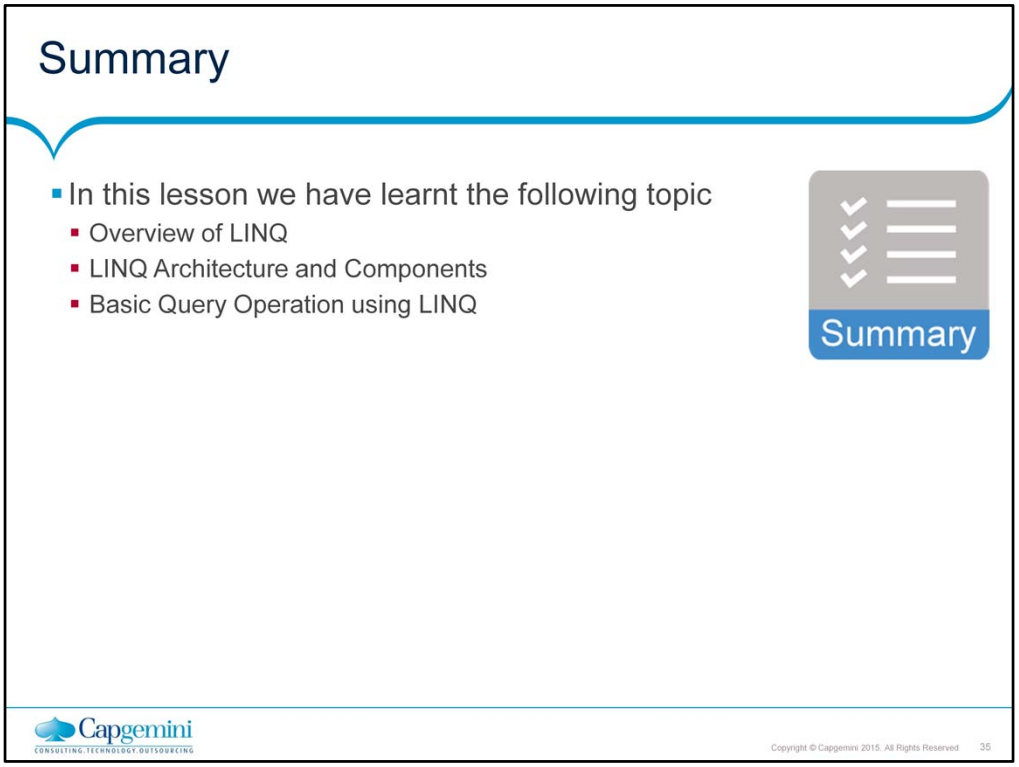
Lab

- Lab Topic

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 34

Add the notes here.



Add the notes here.

Review Question

- Which of the following version of .NET framework introduced LINQ?
  - .NET Framework 2.0
  - .NET Framework 3.5
  - .NET Framework 3.5 SP1
  - .NET Framework 3.0

- LINQ uses programming language syntax to define queries
  - True
  - False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 36

Add the notes here.

Review Question

- LINQ stands for _____
  - Language in Query
  - Language Integrated Query
  - Language Independent Query
  - Language Include Query

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 37

Add the notes here.

LINQ and Entity Framework

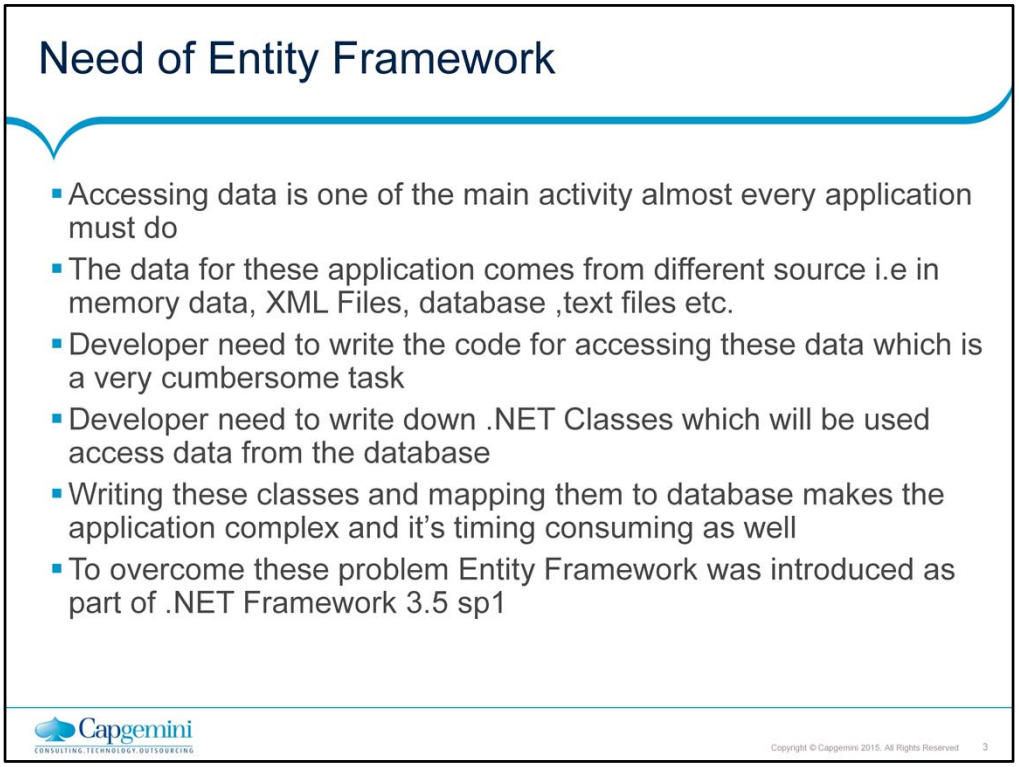
Lesson 02: Introduction to Entity Framework 6.0

Lesson Objectives

- In this lesson we will cover the following
  - Need of Entity Framework
  - Entity Data Model
  - Working with Entities
  - EF Features
  - Back End Support

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

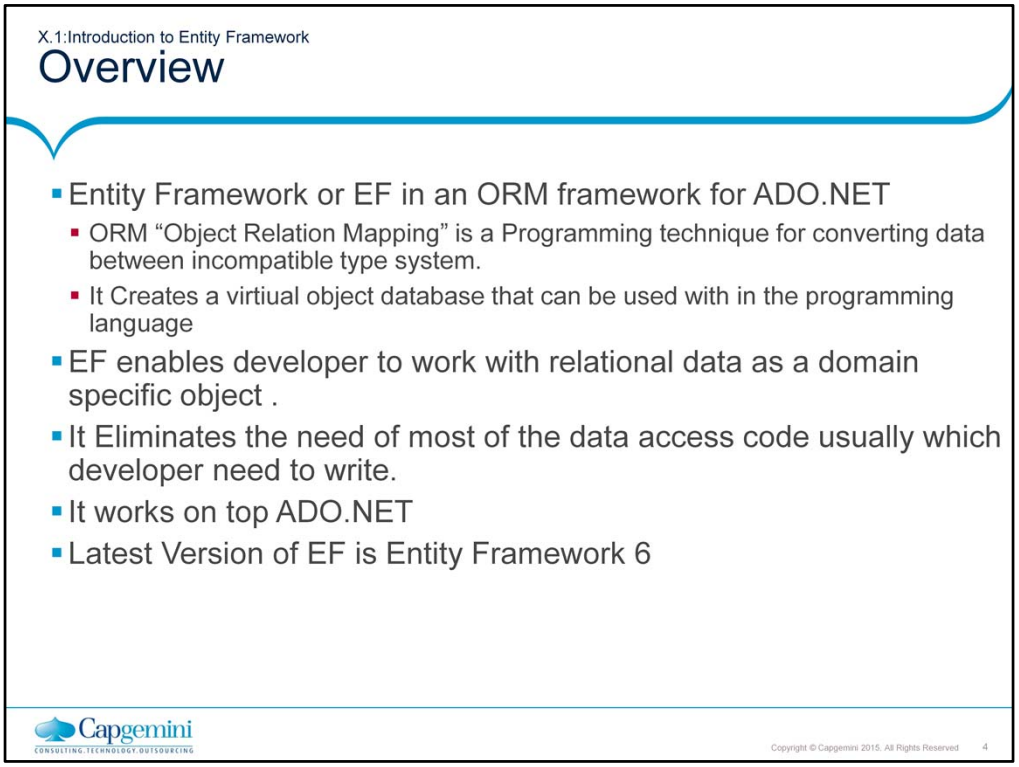


Accessing is one of the main activity almost every application must do .

In .NET a developer use ADO. Net as a data access technology for accomplishing this task , for which they write down data access code with in the application . Writing down these code and managing them is a lot difficult task. Developers uses DataReader and DataSet for accessing data usinf ADO.NET.
Yet, with all the improvements being made to ADO.NET, there was still a disconnect between the application and the back-end database.

Developer has to spend a lot time trying to keep up change being made to database including change in schema or adding a new stored procedure which could potentially break the application flow. Due which developer has to just focus on the data access code instead of the application logic .

To overcome all these problem Microsoft Introduced ADO.NET Entity Framework as a part of .Net Framework 3.5 SP1



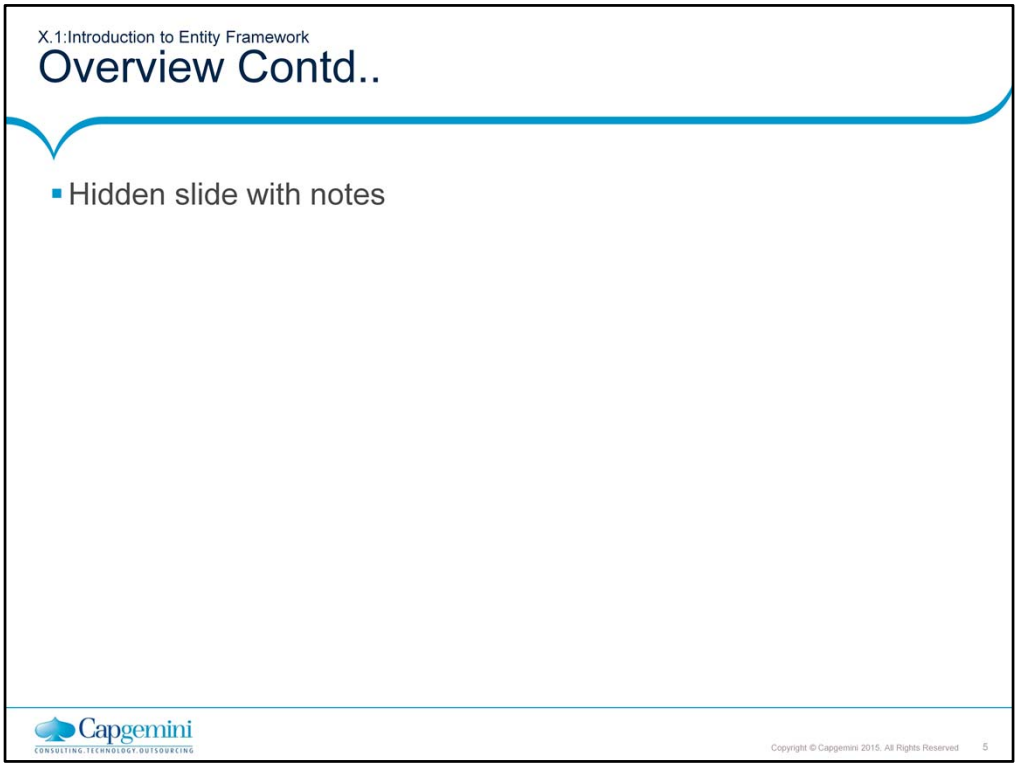
Entity Framework Overview

Microsoft Introduced Entity Framework in .NET Framework 3.5 SP1 and continued it in the next releases of .NET.

The Latest version is Entity Framework 6.0 which has got a lot changes as compared to the previous version of Entity Framework

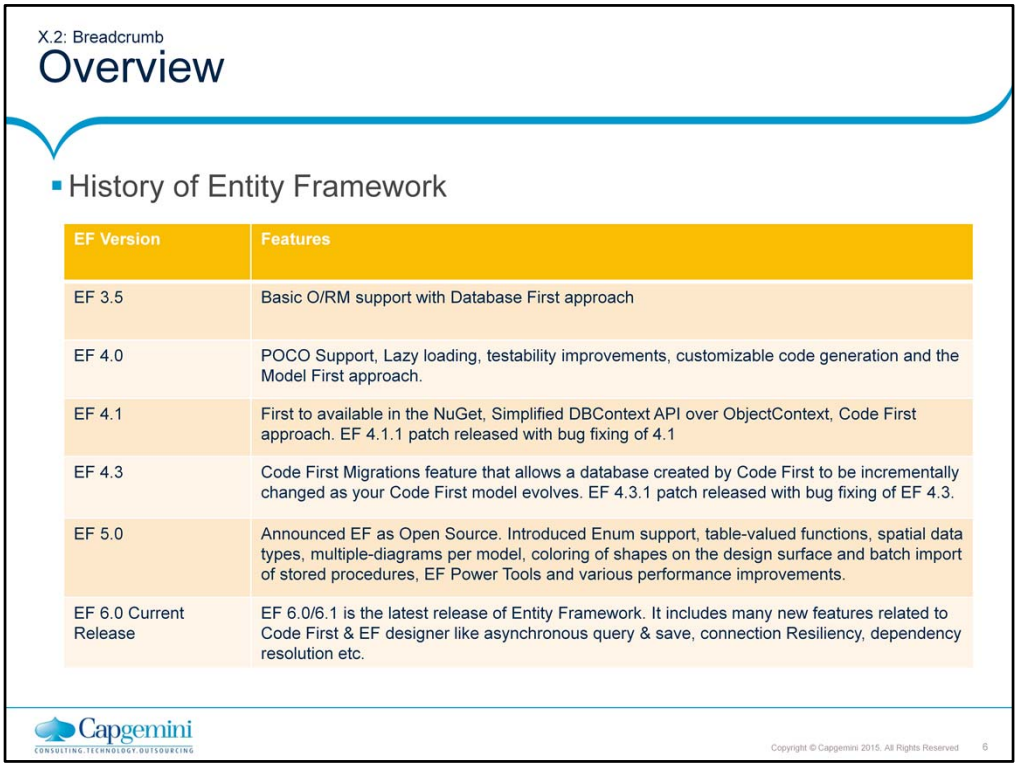
Entity Framework is an Object Relation Mapping Framework which allows a developer to map .NET classes to Database Tables in a Real time Application. It reduces the developer efforts for writing down the data access code.
It work on top of ADO.NET for doing all the database related operations. So the developer doesn't need to write ADO.NET Access code in the application they have to create Entity Data Model which add all the database functionality to the application.

The Microsoft ADO.NET Entity Framework is an Object/Relational Mapping (ORM) framework that enables developers to work with relational data as domain-specific objects, eliminating the need for most of the data access plumbing code that developers usually need to write. Using the Entity Framework, developers issue queries using LINQ, then retrieve and manipulate data as strongly typed objects. The Entity Framework's ORM implementation provides services like change tracking, identity resolution, lazy loading, and query translation so that developers can focus on their application-specific business logic rather than the data access fundamentals.



Entity Framework Overview (Contd..)

ORM is a tool for storing data from domain objects to relational database like MS SQL Server, in an automated way, without much programming. O/RM includes three main parts: Domain class objects, Relational database objects and Mapping information on how domain objects map to relational database objects (tables, views & storedprocedures). ORM allows us to keep our database design separate from our domain class design. This makes the application maintainable and extendable. It also automates standard CRUD operation (Create, Read, Update & Delete) so that the developer doesn't need to write it manually.
A typical ORM tool generates classes for the database interaction for your application


X.2: Breadcrumb

Overview

- History of Entity Framework

EF Version	Features
EF 3.5	Basic O/RM support with Database First approach
EF 4.0	POCO Support, Lazy loading, testability improvements, customizable code generation and the Model First approach.
EF 4.1	First to available in the NuGet, Simplified DBContext API over ObjectContext, Code First approach. EF 4.1.1 patch released with bug fixing of 4.1
EF 4.3	Code First Migrations feature that allows a database created by Code First to be incrementally changed as your Code First model evolves. EF 4.3.1 patch released with bug fixing of EF 4.3.
EF 5.0	Announced EF as Open Source. Introduced Enum support, table-valued functions, spatial data types, multiple-diagrams per model, coloring of shapes on the design surface and batch import of stored procedures, EF Power Tools and various performance improvements.
EF 6.0 Current Release	EF 6.0/6.1 is the latest release of Entity Framework. It includes many new features related to Code First & EF designer like asynchronous query & save, connection Resiliency, dependency resolution etc.

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

 6

The table describes the history of Entity Framework

The above table list all the version of Entity Framework with all the features provided by them with their release

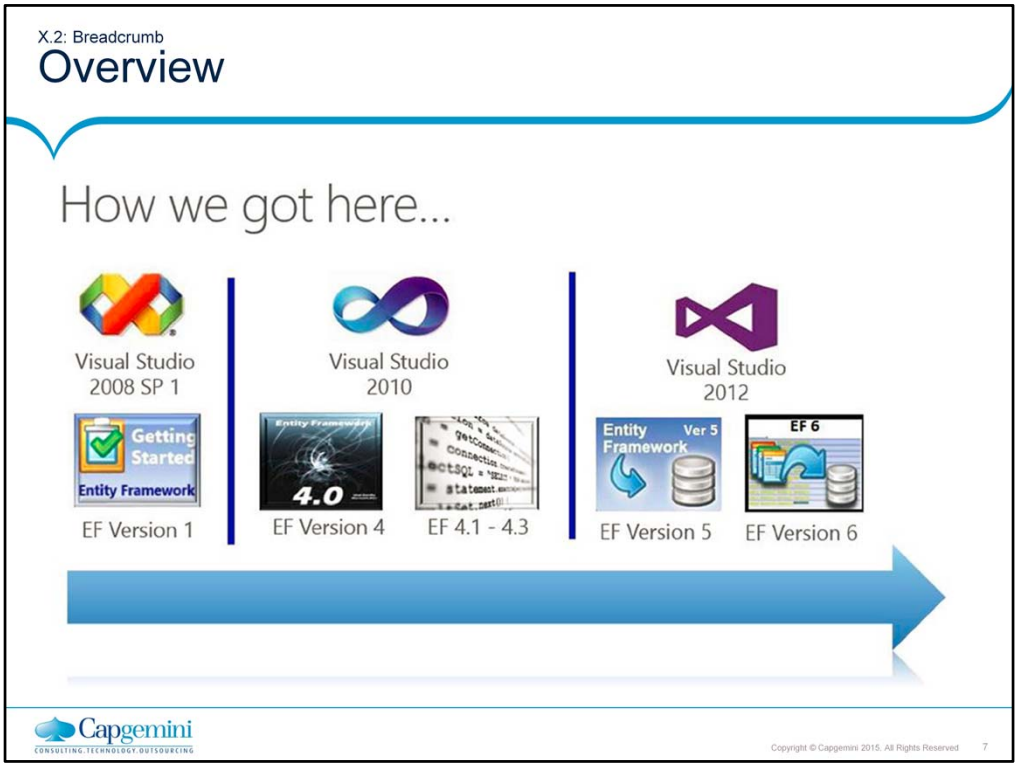
• The first version of Entity Framework was limited, featuring basic ORM support and the ability to implement a single approach known as *Database First.*

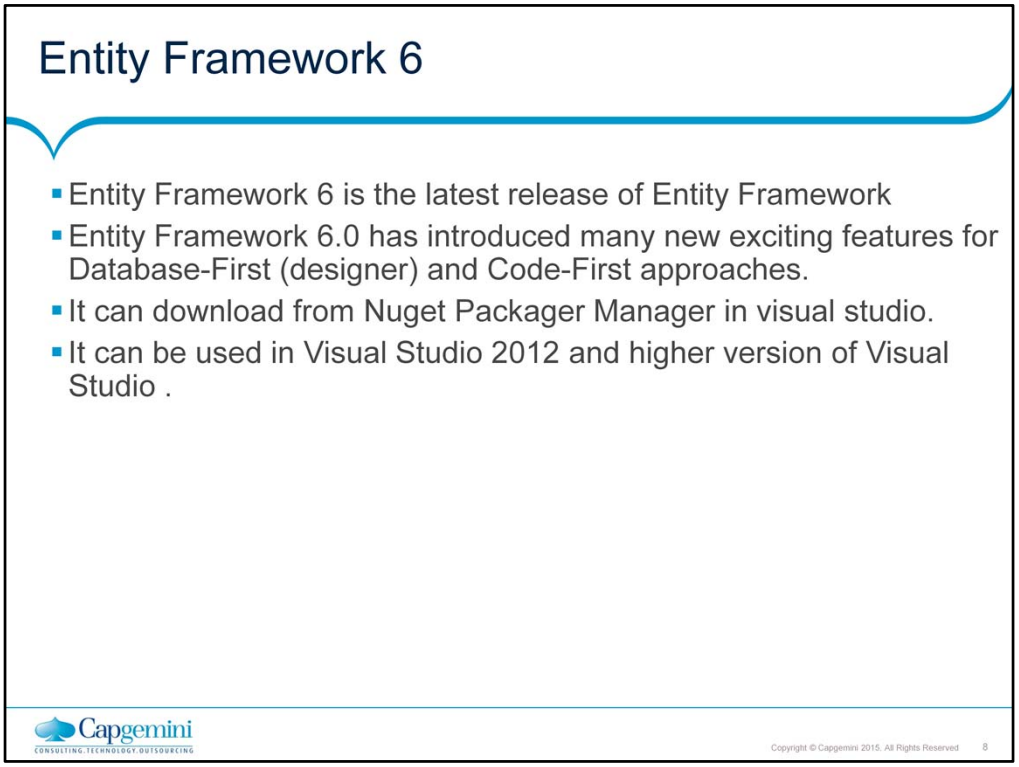
• Version 4 brought us another approach to using Entity Framework: Model First, along with full Plain Old CLR Object (POCO) support and default lazy loading behavior. Soon after, the Entity Framework team released three smaller, or point releases

• Version 4.1 through 4.3, which represented yet another approach to using Entity Framework: Code First.

• Version 5 of Entity Framework coordinated with the release of the .NET 4.5 framework and Visual Studio 2012,delivering significant performance improvements along with support for enums, table value functions, spatial types,the batch import of stored procedures, and deep support with the ASP.NET MVC framework.

• Version 6 of the Entity Framework. Version 6 delivers asynchronous support for querying and updates, stored procedure support for updates in Code First, improved performance, and a long list of new features,


X.2: Breadcrumb
Overview
How we got here...
Visual Studio 2008 SP 1
Visual Studio 2010
Visual Studio 2012
Getting Started
Entity Framework
4.0
Entity Framework
Ver 5
EF 6
EF Version 1
EF Version 4
EF 4.1 - 4.3
EF Version 5
EF Version 6
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
7



Entity Framework 6 is the latest release of Entity Framework providing a lot of new features a compared to the previous version. It is available as part of Visual Studio 2012 and higher version of Visual Studio .

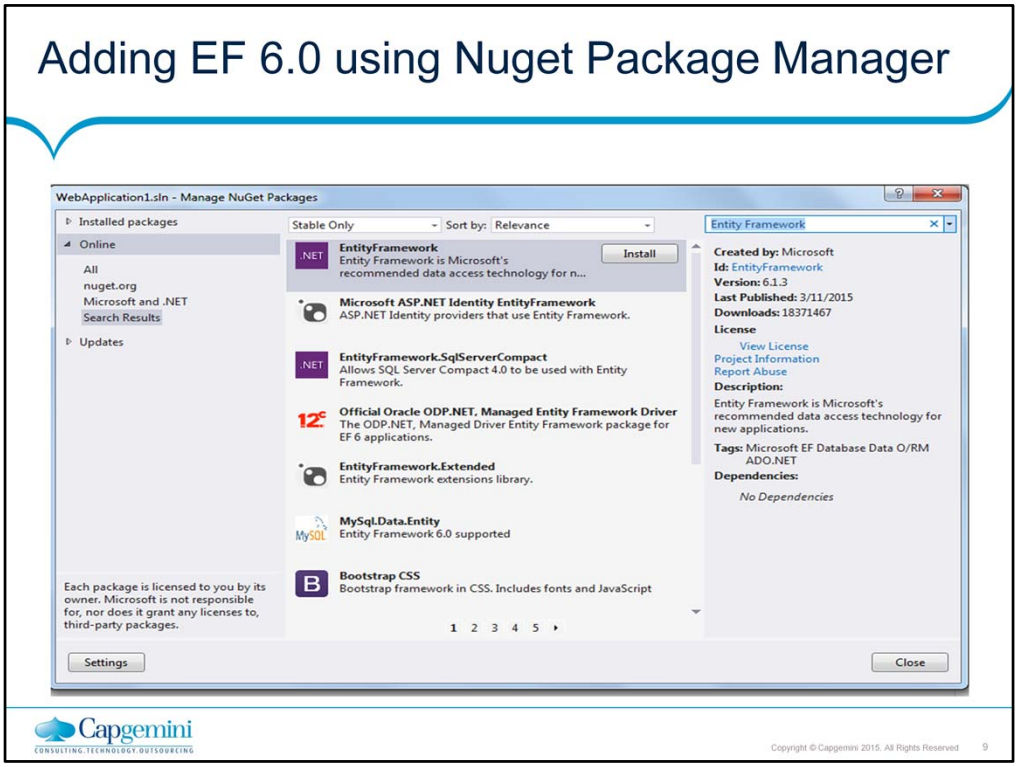
Entity Framework can be added to project using Nuget Package Manager Interface or Nuget Package Manager Console .

What is Nuget ?

NuGet is the package manager for the Microsoft development platform including .NET. The NuGet client tools provide the ability to produce and consume packages. The NuGet Gallery is the central package repository used by all package authors and consumers.

Nuget Package Manager :- Nuget Package Manager provides a dialog using which we search for a specific package and perform installation or uninstallation of packager

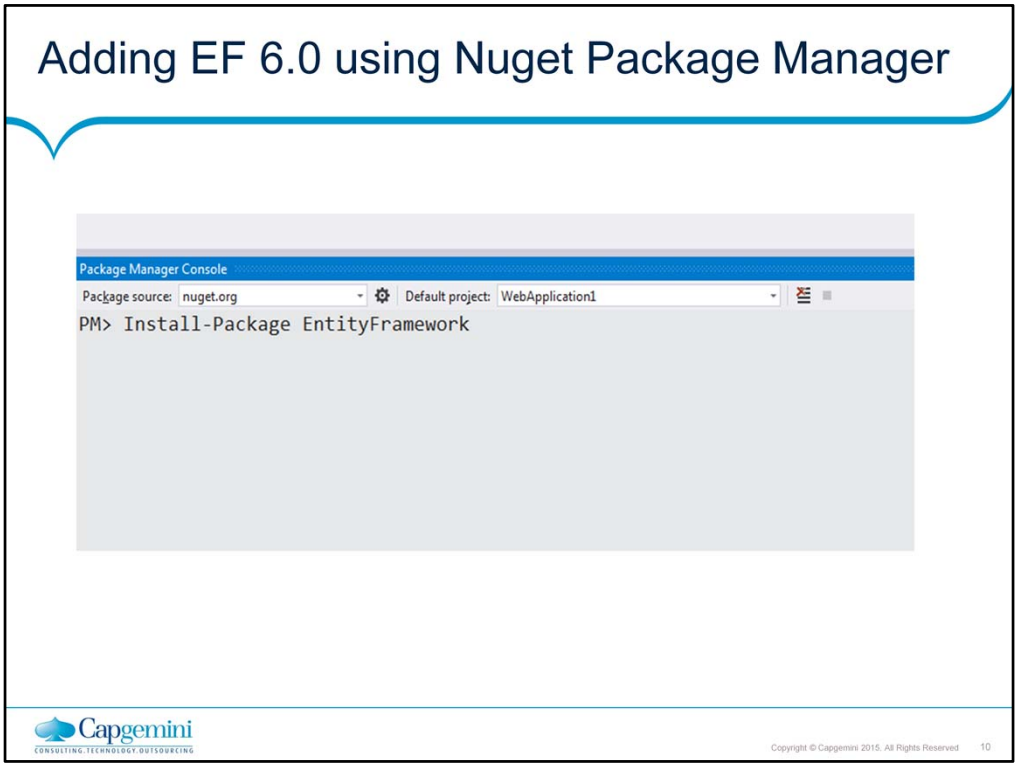
Nuget Package Manager Console :- It is command line tool to add and remove package to a project . It uses power shell command for accomplishing this task.



The above image show the dialog for adding the Entity Framework .

It can be accessed by using the following step :-

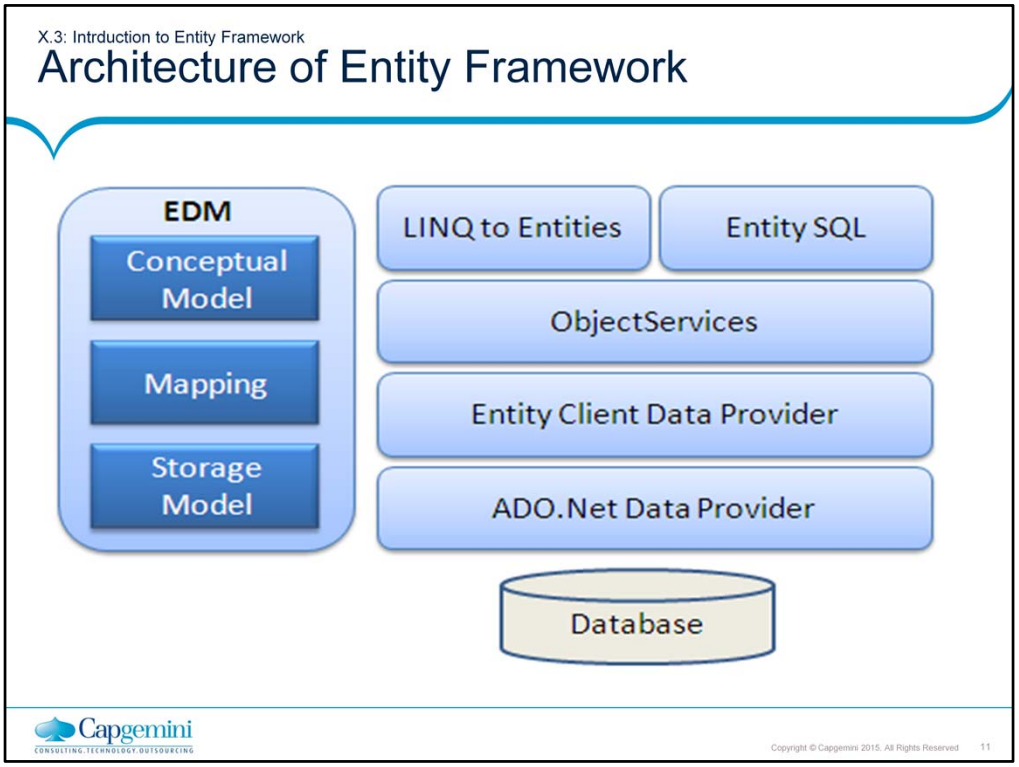
Tool → Nuget Package Manager → Manage Nuget Package for Solution



The above image show the command line tool for adding the Entity Framework .

It can be accessed by using the following step :-

Tool → Nuget Package Manager → Package Manager Console



The above diagram show the architecture and component of Entity Framework

Conceptual Model: The model classes and their relationships. This will be independent of the database table design.

Storage Model: Storage model is the database design model which includes tables, views, stored procedures and their relationships and keys.

Mapping: Mapping contain information about how the conceptual model is mapped to the storage model.

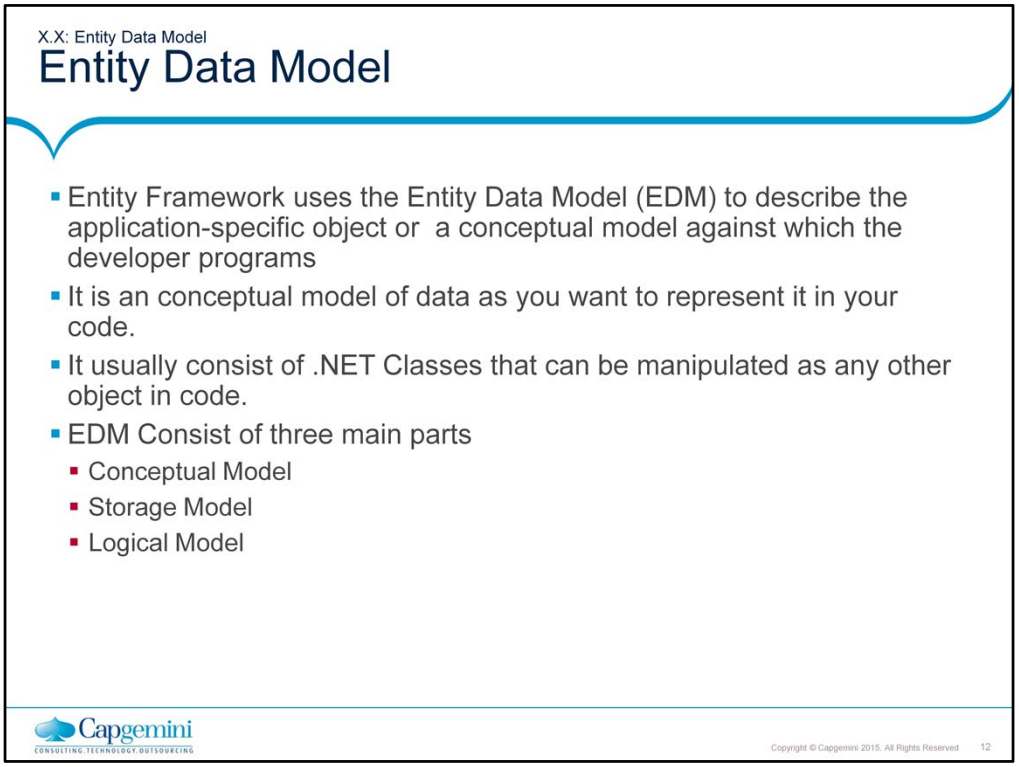
LINQ to Entities: returns entities which are defined in the conceptual model.

Entity SQL: a query language similar to LINQ, but a little more difficult.

Object Service: this provides the entry point for accessing data from a database.

Entity Client Data Provider: The main responsibility of this layer is to convert Linq or Entity SQL queries into SQL queries understood by the underlying database

ADO.Net Data Provider: This layer communicates with the database using standard ADO.Net.



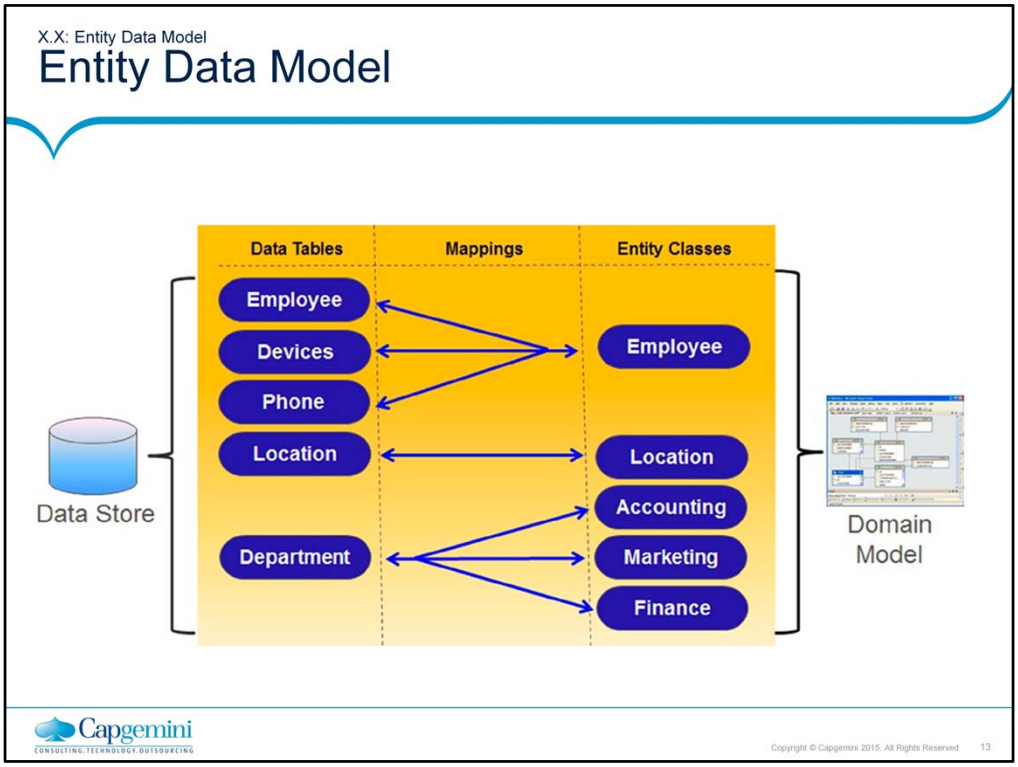
EDM:-

EDM is the foundation of the Entity Framework and is comprised of the three model i.e Conceptual Model, Storage Model, Logical Model. The EDM is the component that describes the overall structure of your business object

Conceptual Model :- The conceptual model contains the classes and their relationships. This will be independent from your database table design.

Storage Model :- Storage model is the database design model which includes tables, view, stored procedures and their relationship and keys.

Logical Model :- Logical Model consist of the information about how the conceptual model is mapped to storage model.



The above diagram show a Entity Data Model

In the above diagram note how the database tables (on the left) do not directly map to the entity classes, which we code against (on the right). Instead, the mapping capabilities built into the Entity Data Model enable the developer to code against a set of entity classes that more closely resemble the problem domain, as opposed to a highly normalized database, designed for performance, scalability, and maintainability.

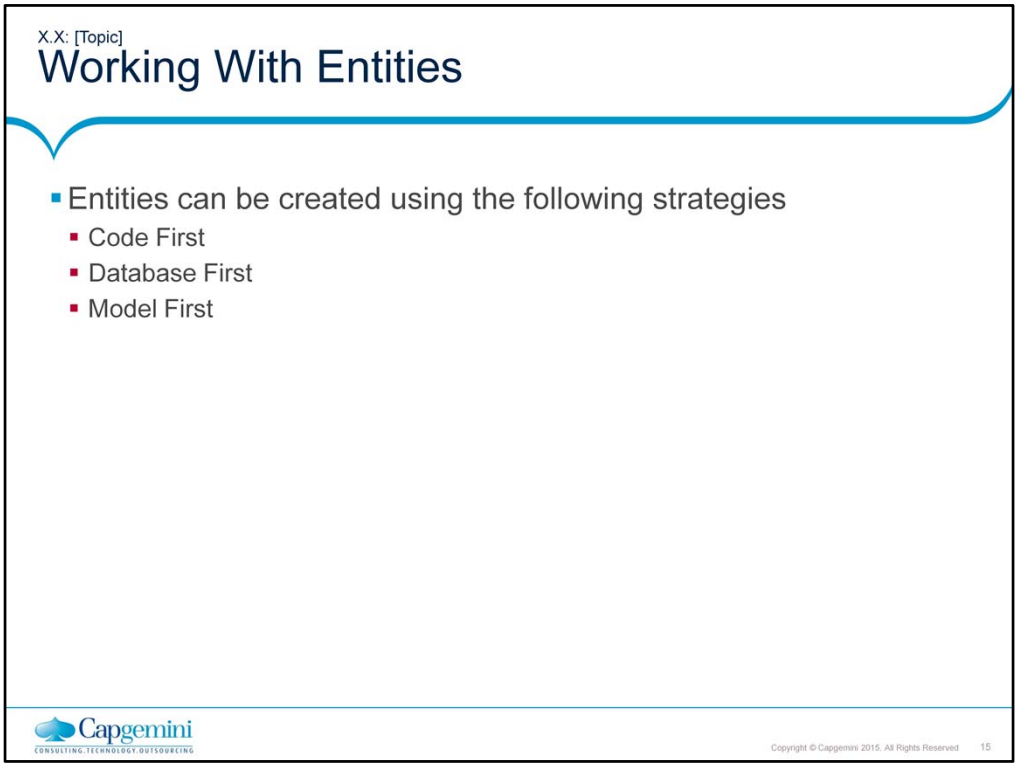
For example, note above how the Employees, Devices, and Phone numbers) are physically stored in three different tables, which from a DBA perspective makes perfect sense. But the developer codes against a single Employee entity class that contains a collection of Devices and Phone Numbers. From a developer and project stakeholder perspective, an employee is a single object, which happens to contain phone numbers and devices.The developer is unaware, and does not care, that the DBA has normalized this employee object into three separate database tables. Once configured, the mapping between the single class and three database tables is abstracted away and handled by the Entity Framework.

Working With Entities

- A key term any Entity framework developer need to know is the term entity
- Entity are like objects eg:-
  - Entities have a know type
  - Entities have properties and these properties can hold scalar values
  - Entity properties can hold to references to other entites
  - Each entity has a distinct identity
- Entities are extremely flexible
- Entities can have relationships between them

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

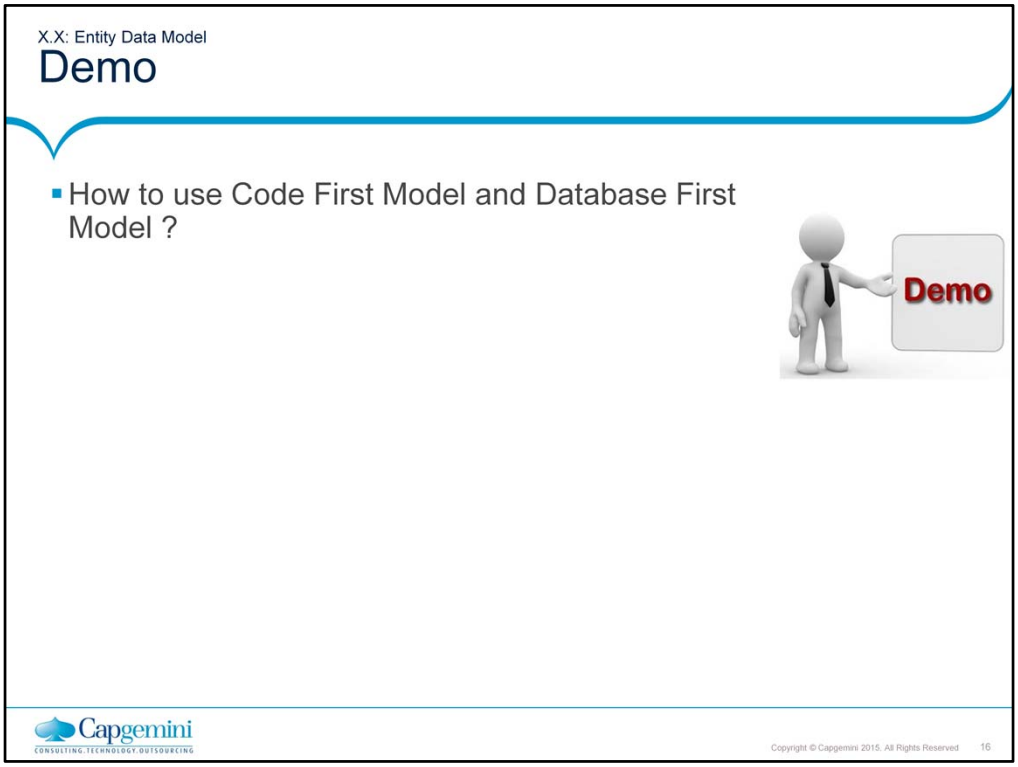
 14



Code First :- In the Code First approach, you avoid working with visual model designer (EDMX) completely. You write your POCO classes first and then create database from these POCO classes. Developers who follow the path of Domain-Driven Design (DDD) principles, prefer to begin by coding their domain classes first and then generating the database required to persist their data.

Database First :- In the Database First Approach ,you can use to existing database to entity data model and work on it.

Model First :- In the Model First approach, you create Entities, relationships, and inheritance hierarchies directly on the design surface of EDMX and then generate database from your model.So, in the Model First approach, add new ADO.NET Entity Data Model and select **Empty EF Designer model** in Entity Data Model Wizard.



Add the notes here.
Demonstrate Code First and Database First Model

Steps for Creating EDM using Database First Model.

•**Project -> Add New Item…**

•Select **Data** from the left menu and then **ADO.NET Entity Data Model**

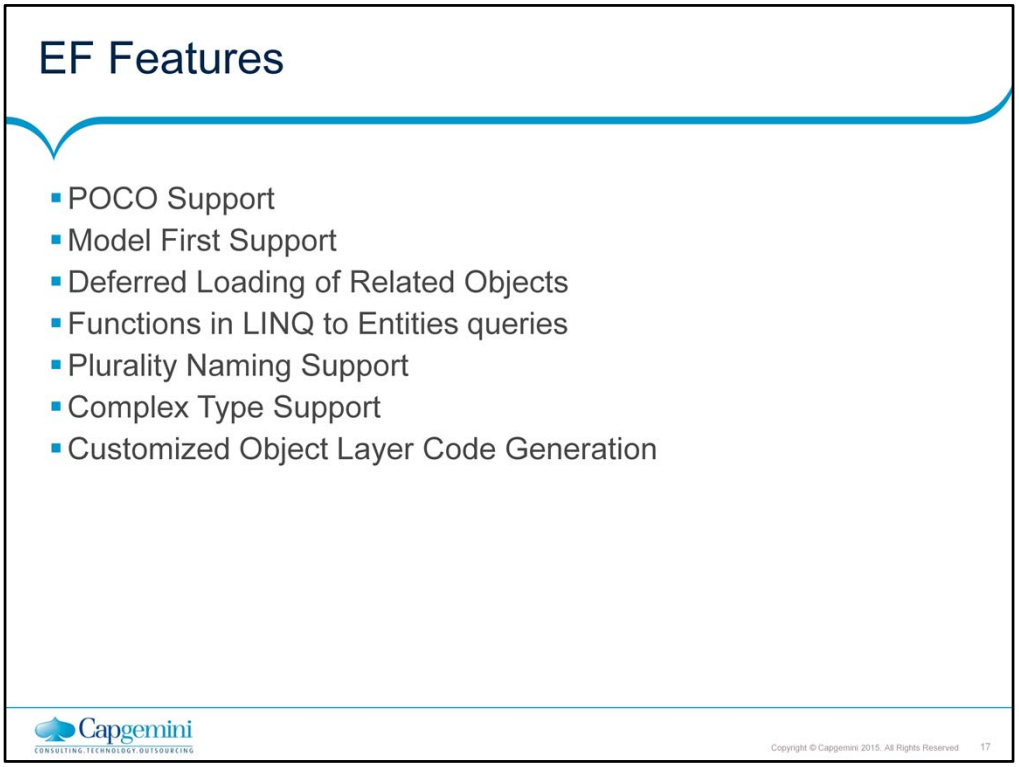
•Enter **BloggingModel** as the name and click **OK**

•This launches the **Entity Data Model Wizard**

•Select **Generate from Database** and click **Next**

•Select the connection to the database you created in the first section, enter **BloggingContext** as the name of the connection string and click **Next**

•Click the checkbox next to ‘Tables’ to import all tables and click ‘Finish’



EF Features
Poco Support :- One of the more powerful new features of the Entity Framework is the ability to add and use your own custom data classes in conjunction with your data model. This is accomplished by using CLR objects, commonly known as “POCO” (Plain Old CLR Objects). The added benefit comes in the form of not needing to make additional modifications to the data classes. This is also called persistence-ignorance.

Model First Support :- It Allows the developer to create a conceptual model using the create database wizard. It creates the database based on the conceptual model specified.

Deferred Loading of Related Objects :- Deferred Loading also know as Lazy loading . It helps the query result to be shaped by composing queries that explicitly navigate the relationship via the navigation properties

LINQ to Entities :- Adds the Linq query capabilities .Using this you can access data from the entity using LINQ

Plurality Naming :- It Pluralize the name of entities while creating them . This adds a proper Naming Convention which helps developer to access using code.

Complex Type :- Complex types are like entities in that they consist of a scalar property or one or more complex type properties. Thus,complex types are non-scalar properties of entity types that enable scalar properties to be organized within entities.

Object Layer Code Generation :- Object Layer Code is generated by default .It allows developer to add text templates to a project . By using custom text template , the EDM will generate the object context and entity classes.

EF 6 Features for Database First

- Connection resiliency
- Asynchronous query and save
- Code-based configuration
- Database command logging
- Database command interception
- Dependency Resolution
- DbSet.AddRange/RemoveRange
- Better Transaction Support
- Pluggable pluralisation and singularization service
- Testability improvements
- Creating context with an open connection
- Improved performance and warm-up time

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

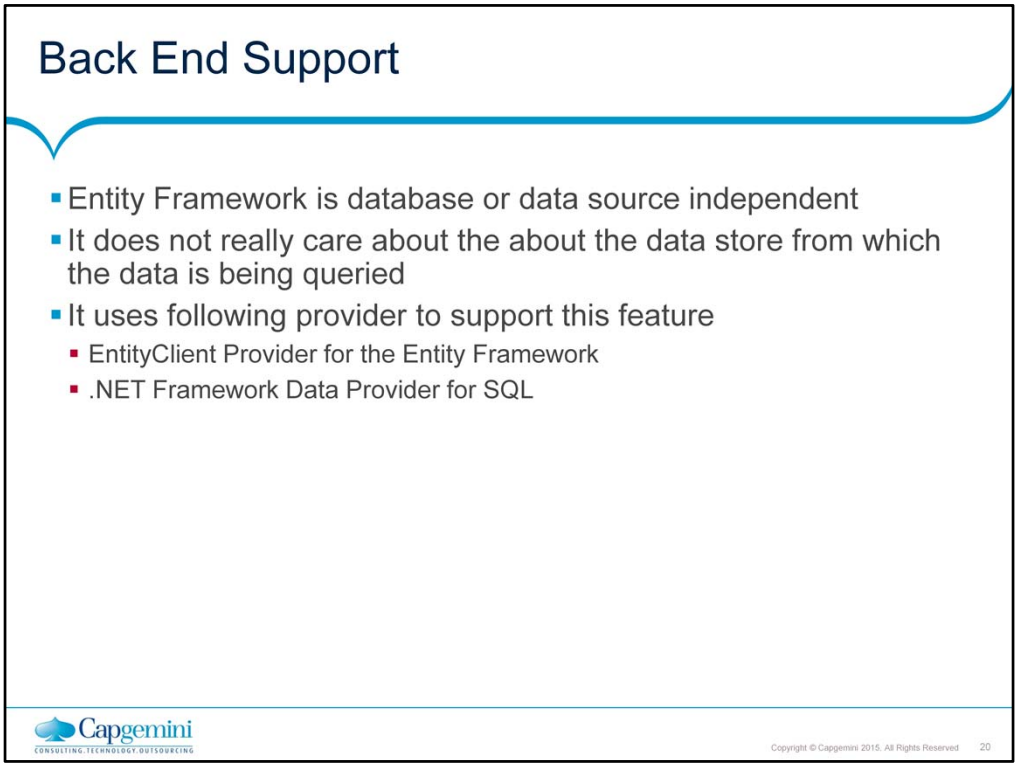
 18

EF 6 Features for Code-First

- Custom conventions
- Insert, update & delete stored procedures for entity CUD operation
- Index attribute (EF 6.1)
- Multiple context per database
- Nested entity types
- Custom migration operations
- Configurable migration history table

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

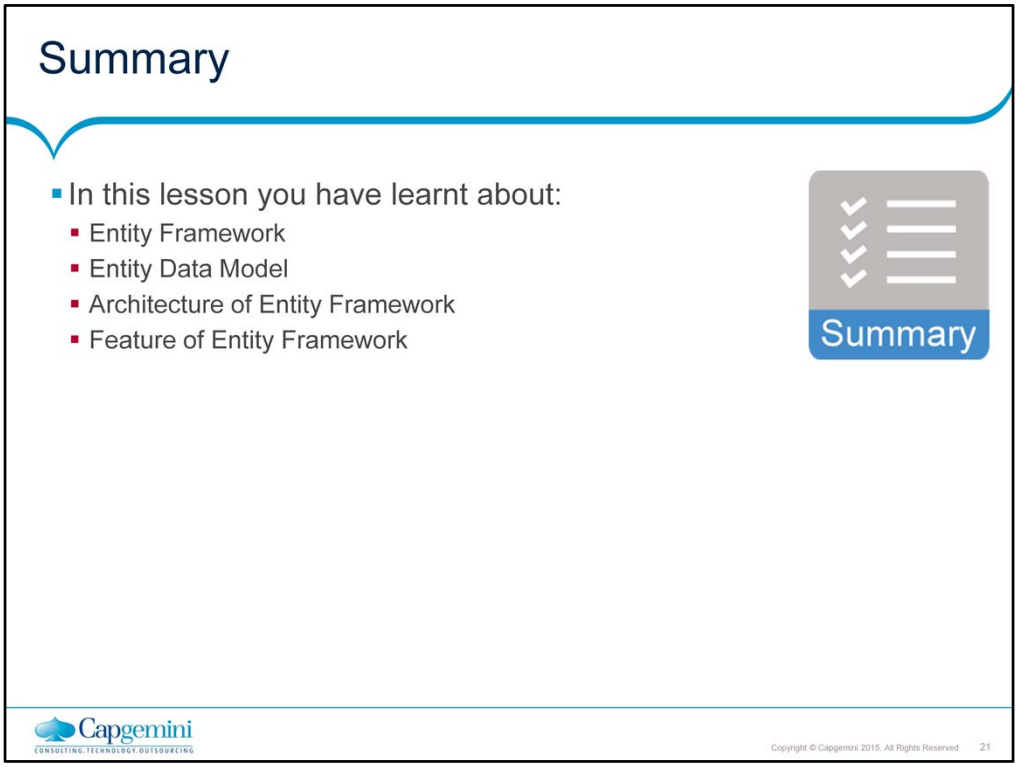
 19



The great thing about the Entity Framework is that in essence it does not really care about the data store from which the data is being queried. Neither the type of database nor the schema itself is completely unknown to the Entity Framework, and they will have no impact on your model.

The Entity Framework ships with two providers:
• EntityClient Provider for the Entity Framework*:* Used by Entity Framework applications to access data described in the EDM. This provider uses the .NET Framework Data Provider for SQL Server (SqlClient) to access a SQL Serverdatabase.
• NET Framework Data Provider for SQL Server (SqlClient) for the Entity Framework: Supports the Entity Framework for use with a SQL Server database.

As Being database or data source independent we can create custom providers to access other database like Oracle, MySql, PostgreSQL, DB2 etc



Add the notes here.

Review Question

- Which of the following EDM consist of ?
  - Conceptual Model
  - Storage Model
  - Code Model
  - Logic Model
- Entity Framework works on top of ADO.NET?
  - True
  - False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

Add the notes here.

Review Question

- The _____ contains the classes and their relationships.
  - Logic Model
  - Storage Model
  - Conceptual Model

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

Add the notes here.

LINQ and Entity Framework

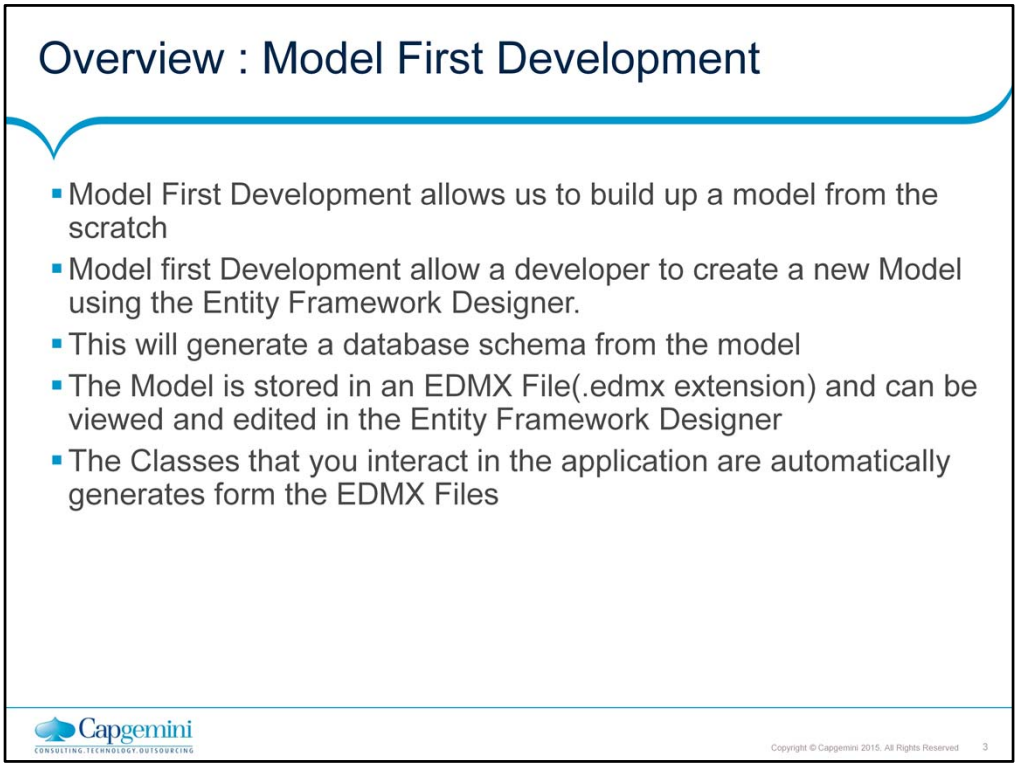
Lesson 03: EDM Tool Enhancement

Lesson Objectives

- In this lesson we will cover the following
  - Model First Development
  - Pluralization
  - Complex Types

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



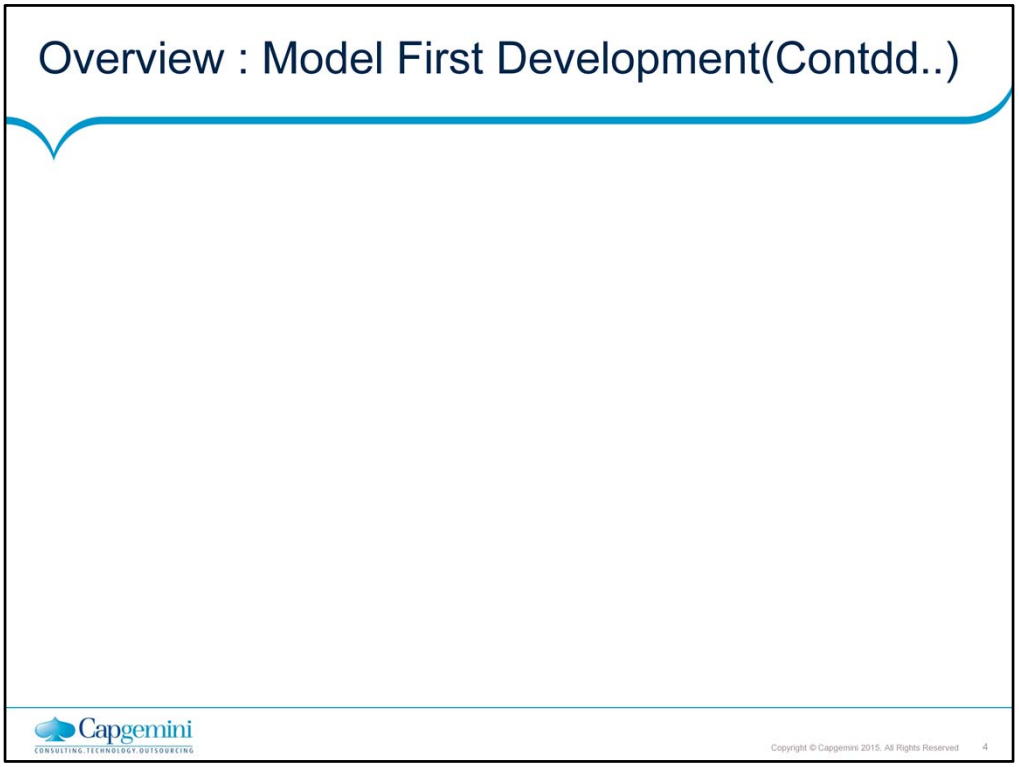
Model First Development allows to build up a Model from scratch . It allows you to create Entities, relationships, and inheritance hierarchies directly on the design surface of EDMX and then generate database from your model.

With the Entity Designer, we can define entities which are abstractions representing the objects in application domain. This is where we create a Entity Data Model (EDM) and is driven by an XML grammar from a model file extension of .EDMX

The Model First Approach was provided by Microsoft because it was one of the most often requested features by .Net Developers and System Designers. It is a more natural way to work compared to the **Database First** approach especially when designing the flow of data in the initial stages of a new project. Developers requested the flexibilty of creating the conceptual model first instead of going through the burden of outlining the project logic and then trying to design a database which would accomodate for all the data and information storage within the project. Using the **Model First** approach a developer can start working with the model of the databse and creating entities which make logical sense irrelevant of how they will actually be stored in the database in terms of tables.

From the model created, Visual Studio can then generate SQL statements referred to as *Data Defenition Language (DDL)* which the developer can use to execute on MS SQL in order to create the Database schema based on the designed model.

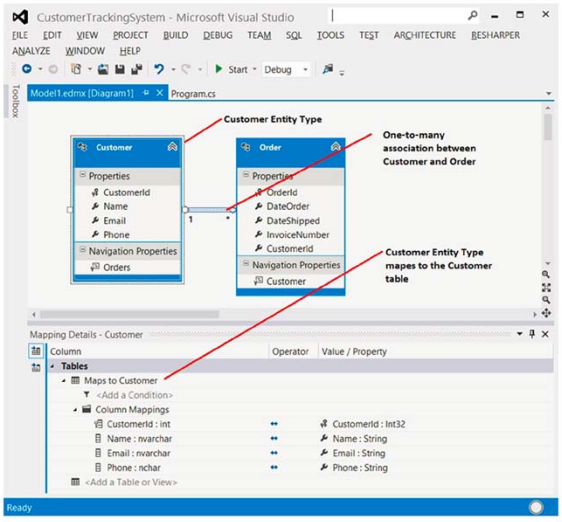
As an example, we might create an Order entity that will represent instances of orders and their details such as who placed the order, on what date and so on. Then we might create related entities to hold additional information such as the items in the order and their cost and quantities. The modeling experience gives a natural way to build up a description of the data in application visually, playing around with it and changing it easily The real power of this comes though when we want to turn the model into reality. That's because EF can take the EDM and generate two things from it, the Data Definition Language (DDL) to create a database as the model's concrete representation and data access code to manipulate it.



From the model created, Visual Studio can then generate SQL statements referred to as *Data Defenition Language (DDL)* which the developer can use to execute on MS SQL in order to create the Database schema based on the designed model.

As an example, we might create an Order entity that will represent instances of orders and their details such as who placed the order, on what date and so on. Then we might create related entities to hold additional information such as the items in the order and their cost and quantities. The modeling experience gives a natural way to build up a description of the data in application visually, playing around with it and changing it easily The real power of this comes though when we want to turn the model into reality. That's because EF can take the EDM and generate two things from it, the Data Definition Language (DDL) to create a database as the model's concrete representation and data access code to manipulate it.

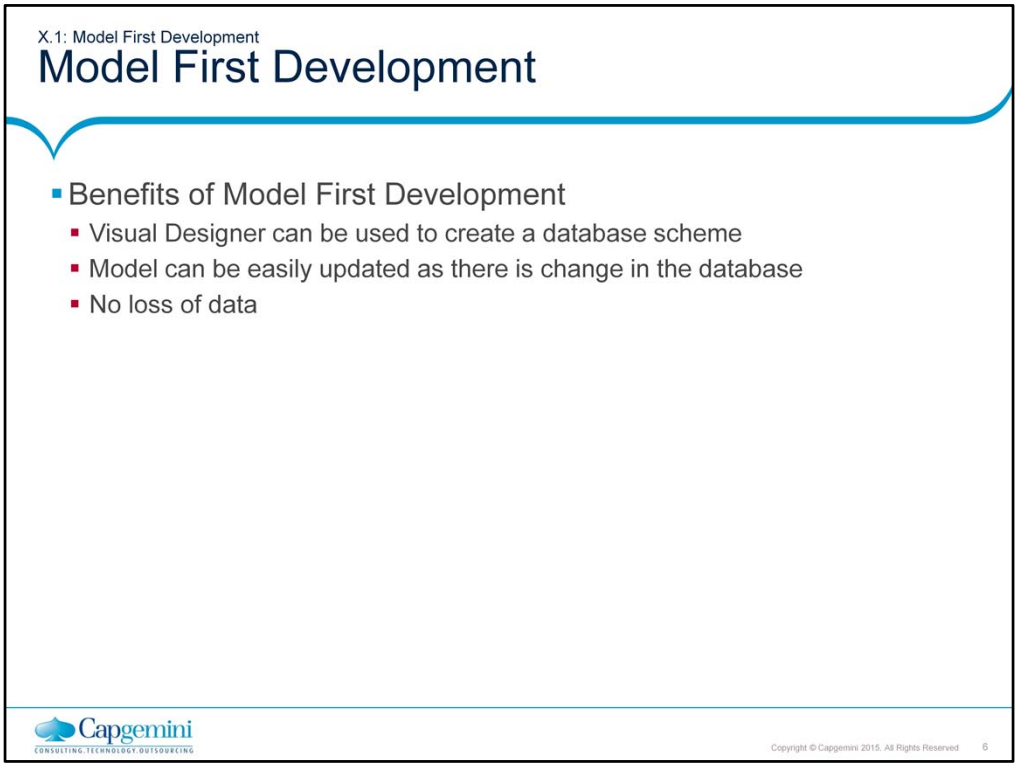
Model First Development (Contd..)



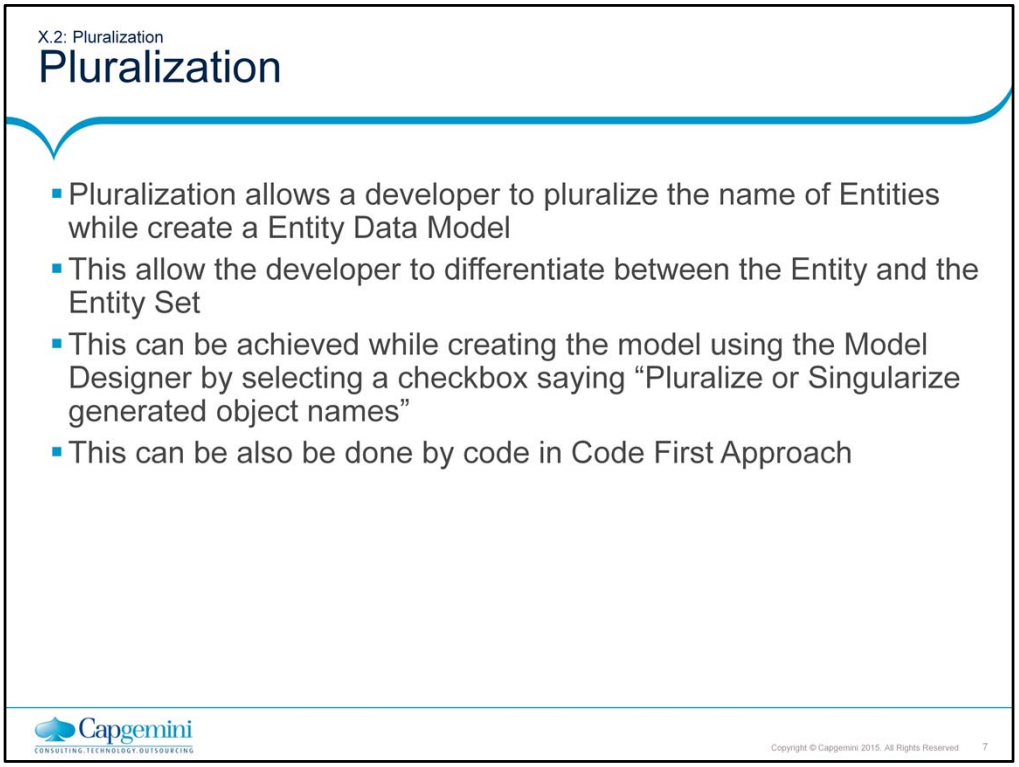
The above image shows the EDMX designer in Visual Studio .

The Image two entities in the designer i.e Customer and Order .

It show all the fields of the Entities and the relationship between them .

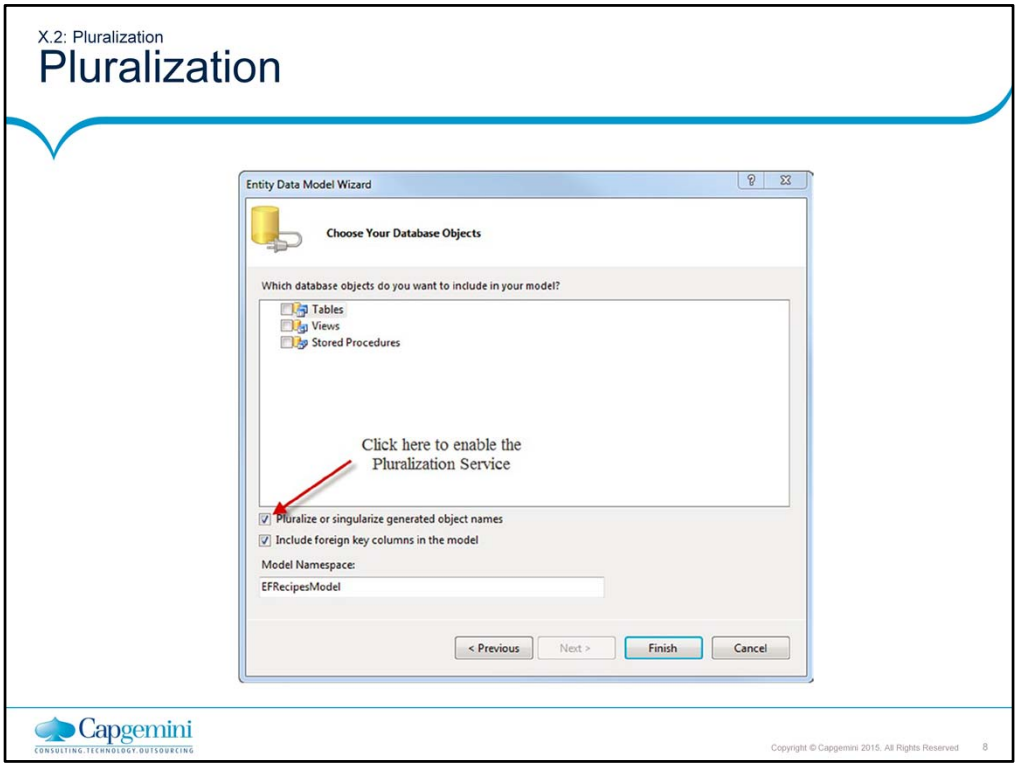


Add the notes here.



Pluralization :-

Pluralization allow a developer to Pluralize the name of Entity to represent an Entity Set.



The above screenshot show how to enable Pluralization in Model First and Database First approach

X.X: Model First Development

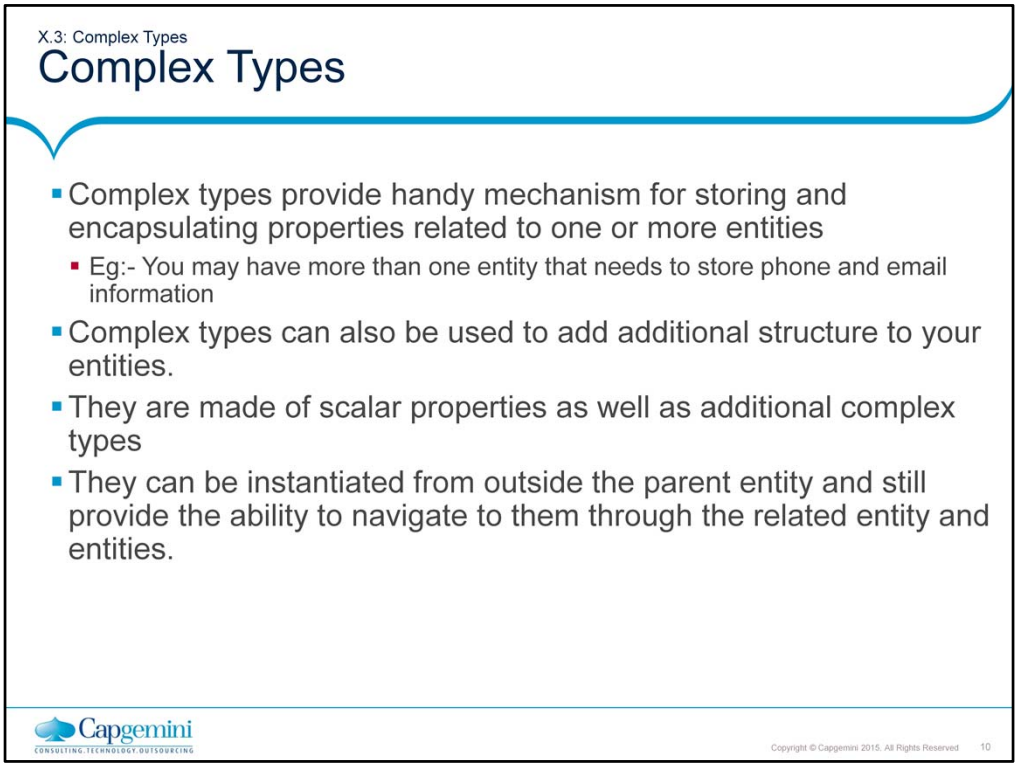
Demo

- Demo of Implementing Model First Development approach in Console Application.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

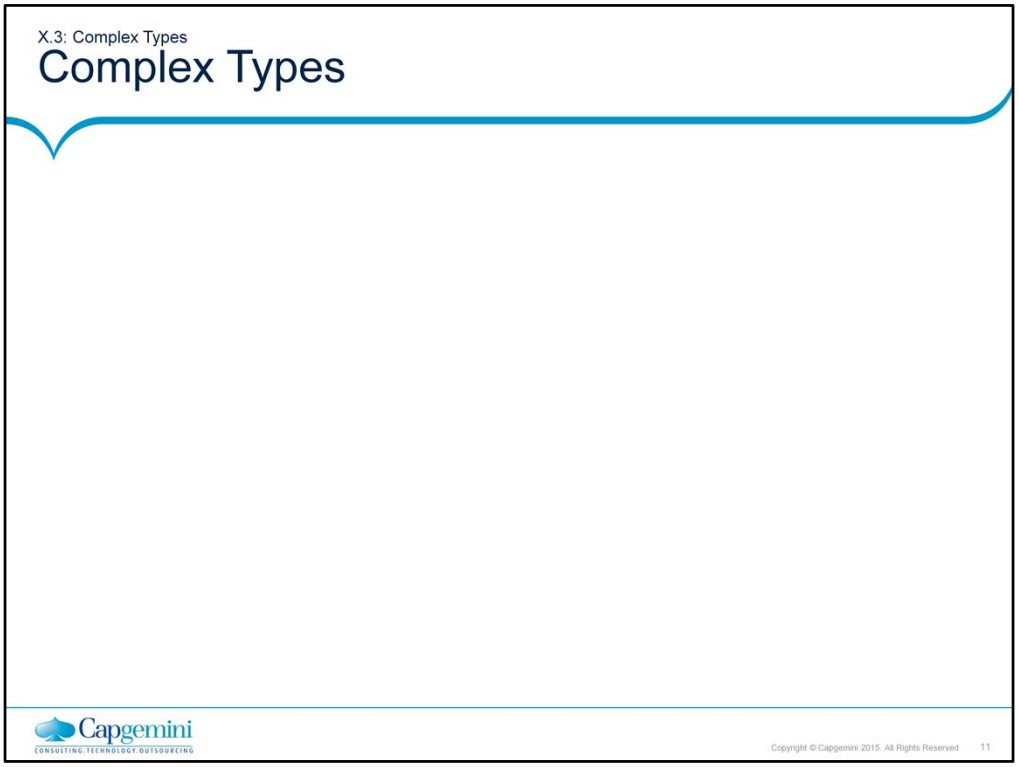
 9

Add the notes here.



Complex Type:-

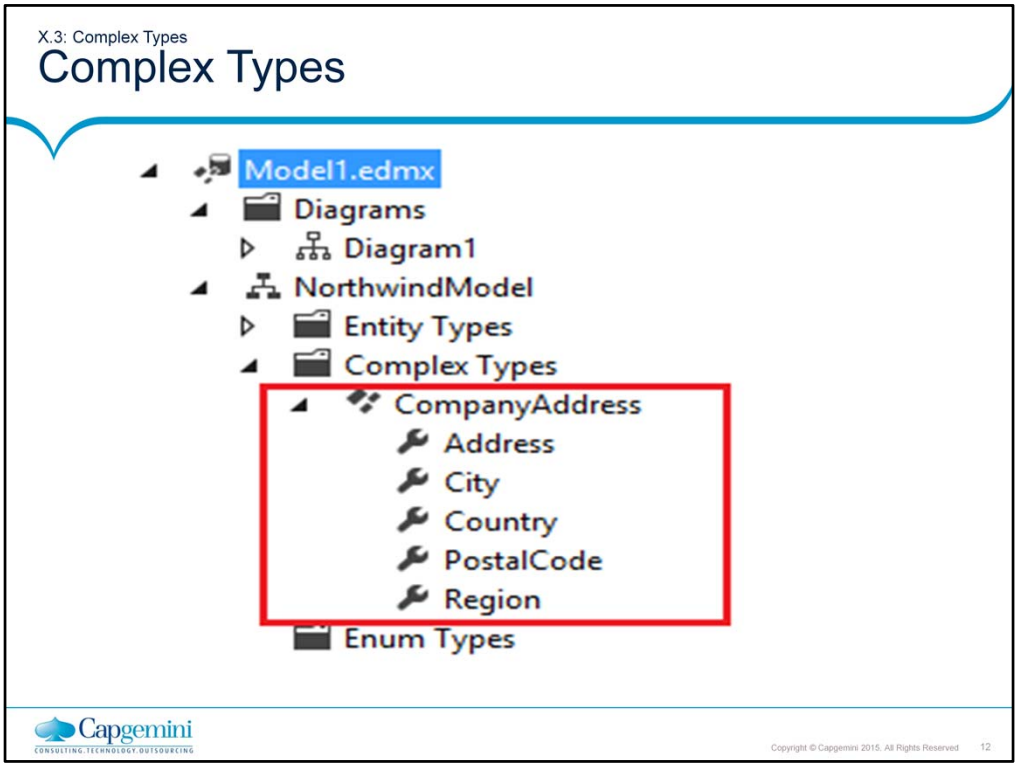
•Complex types are non-scalar properties of entity types that enable scalar properties to be organized within entities. Like entities, complex types consist of scalar properties or other complex type properties.
•Complex types allow you to group several properties into a single type for a property on an entity.
•A complex type can contain scalar properties or other complex types, but they cannot have navigation properties or entity collections.
•A complex type cannot be an entity key. Complex types are not tracked on their own in an object context.
•A property whose type is a complex type cannot be null. When you work with entities with complex type properties, you have to be mindful of this rule.
•Occasionally, when the value of a complex type property is unimportant for a particular operation, you may need to create a dummy value for the property so that it has some nonnull value.
•When you modify any field in complex type property, the property is marked as changed by Entity Framework, and an update statement will be generated that will update all of the fields of the complex type property.
•Complex type can also be with POCO (Plain Old CLR Objects).



Complex Type:-

When you work with objects that represent complex types, be aware of the following:

•Complex types do not have keys and therefore cannot exist independently. Complex types can only exist as properties of entity types or other complex types.
•Complex types cannot participate in associations and cannot contain navigation properties.
•Complex type properties cannot be **null**. An **InvalidOperationException** occurs when **DbContext.SaveChanges** is called and a null complex object is encountered. Scalar properties of complex objects can be **null**.
•Complex types cannot inherit from other complex types.
•You must define the complex type as a **class**.
•EF detects changes to members on a complex type object when **DbContext.DetectChanges** is called. Entity Framework calls **DetectChanges** automatically when the following members are called: **DbSet.Find**, **DbSet.Local**, **DbSet.Remove**, **DbSet.Add**, **DbSet.Attach**, **Db Context.SaveChanges**,**DbContext.GetValidationErrors**, **DbContext.Entry**, **DbC hangeTracker.Entries**.



The above screenshot shows a complex type CompanyAddress as a part of Model .

As you can see it is consisting of other types which make up CompanyAddress

X.X: Complex Types

Demo

- Demo for Creating and Using Complex Type in Entity Framework

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Add the notes here.

X.X: [Topic]

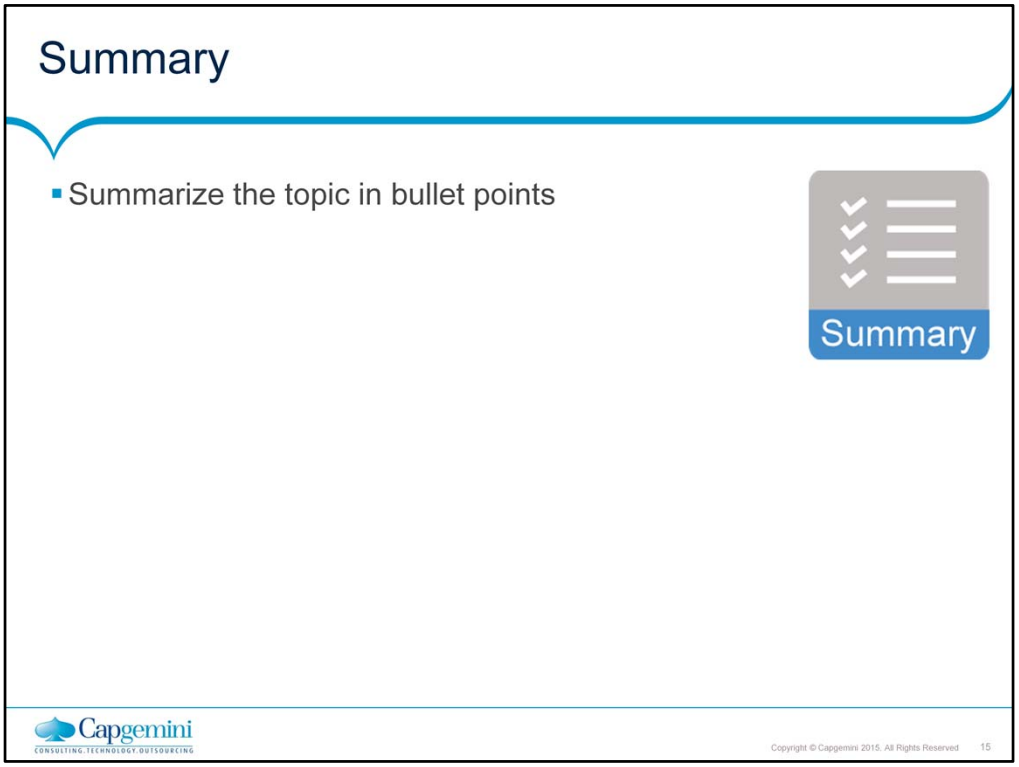
Lab

- Lab Topic

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Add the notes here.



Add the notes here.

Review Question

- Which of the following allows the development of model from scratch using model designer?
  - Design First
  - Code First
  - Model First
  - Database First

- Complex type cannot be used to add additional structure to the model
  - True
  - False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

Add the notes here.

Review Question

- _____allow a developer to create a new Model using the Entity Framework Designer.
  - Code First Development
  - Designer First Development
  - Model First Development

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

Add the notes here.

LINQ and Entity Framework

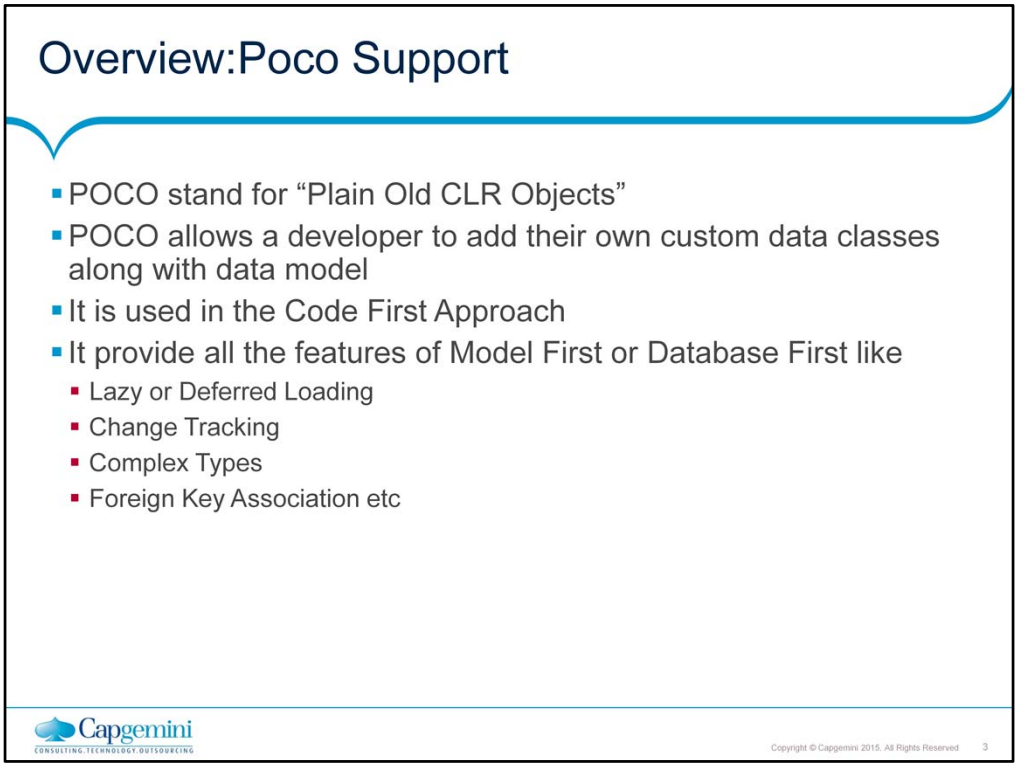
Lesson 04: EF Runtime Features

Lesson Objectives

- In this lesson we will cover the following
  - POCO Support
  - Lazy or Deferred Loading
  - Foreign Keys

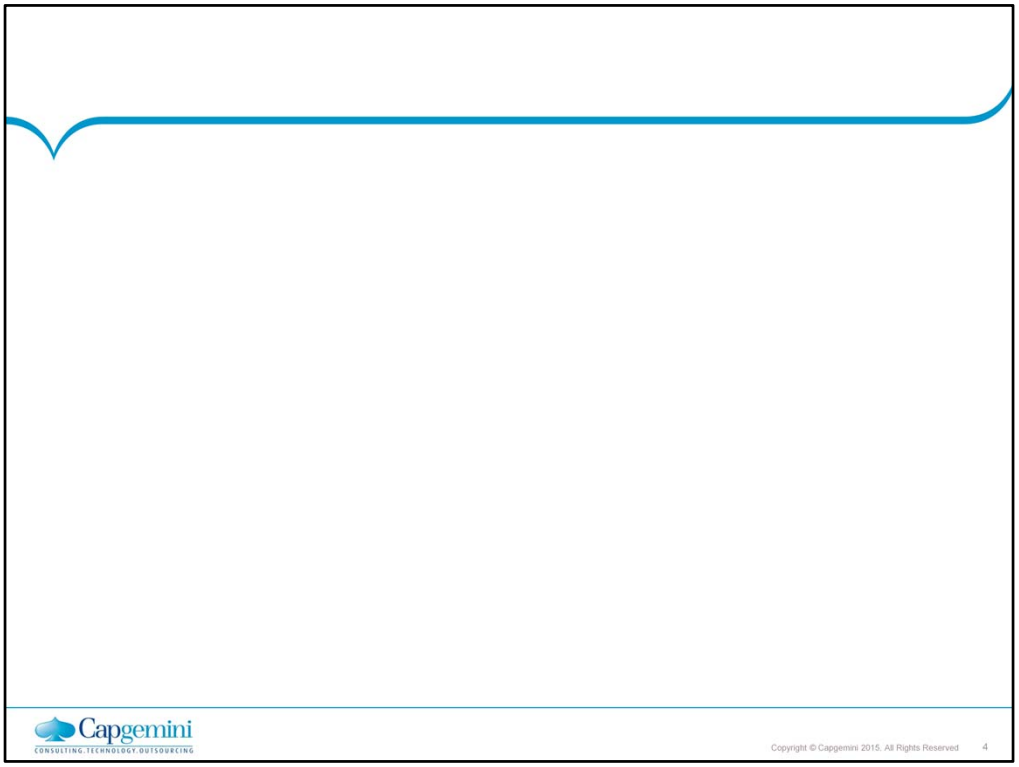
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



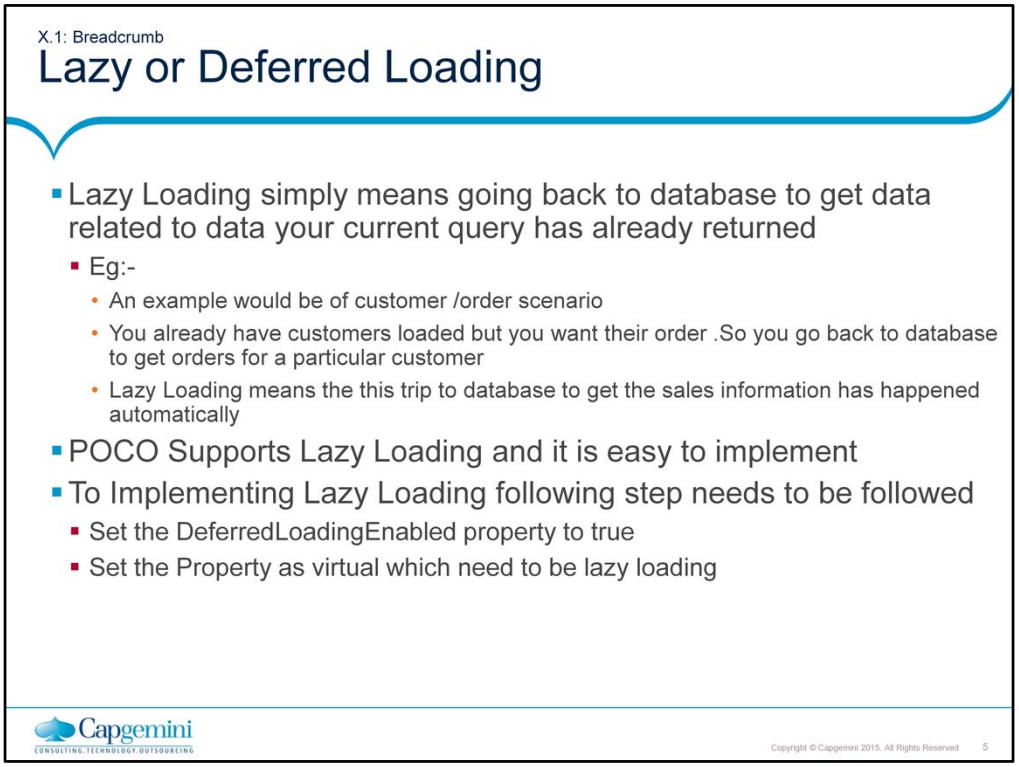
POCO Suport

•One of the more powerful new features of the Entity Framework is the ability to add and use your own custom data classes in conjunction with your data model.
•This is accomplished by using CLR objects, commonly known as “POCO” (Plain Old CLR Objects).
•The added benefit comes in the form of not needing to make additional modifications to the data classes.
•This is also called persistence-ignorance.
•The flexibility of extending these partial classes means more control over the core entity object functionality.
•This is a huge advantage as developers can now leverage and preserve valuable customizations and business logic, which they might not have been able to do previously.



The code-only approach simply requires that you create POCO classes that contain the same
structure as the database schema you want to map to, such as the following Contact class

```
public class Contact
{
public Contact() { }
public int ContactID { get; set; }
public bool NameStyle { get; set; }
public string Title { get; set; }
public string FirstName { get; set; }
public string MiddleName { get; set; }
public string LastName { get; set; }
public string Suffix { get; set; }
public string EmailAddress { get; set; }
public int EmailPromotion { get; set; }
public string Phone { get; set; }
public string PasswordHash { get; set; }
public string PasswordSalt { get; set; }
public Guid rowguid { get; set; }
public DateTime ModifiedDate { get; set; }
public ICollection<Employee> Employees { get; set; }
}
```



Lazy Loading :-

Lazy loading is the process whereby an entity or collection of entities is automatically loaded from the database the first time that a property referring to the entity/entities is accessed. When using POCO entity types, lazy loading is achieved by creating instances of derived proxy types and then overriding virtual properties to add the loading hook.

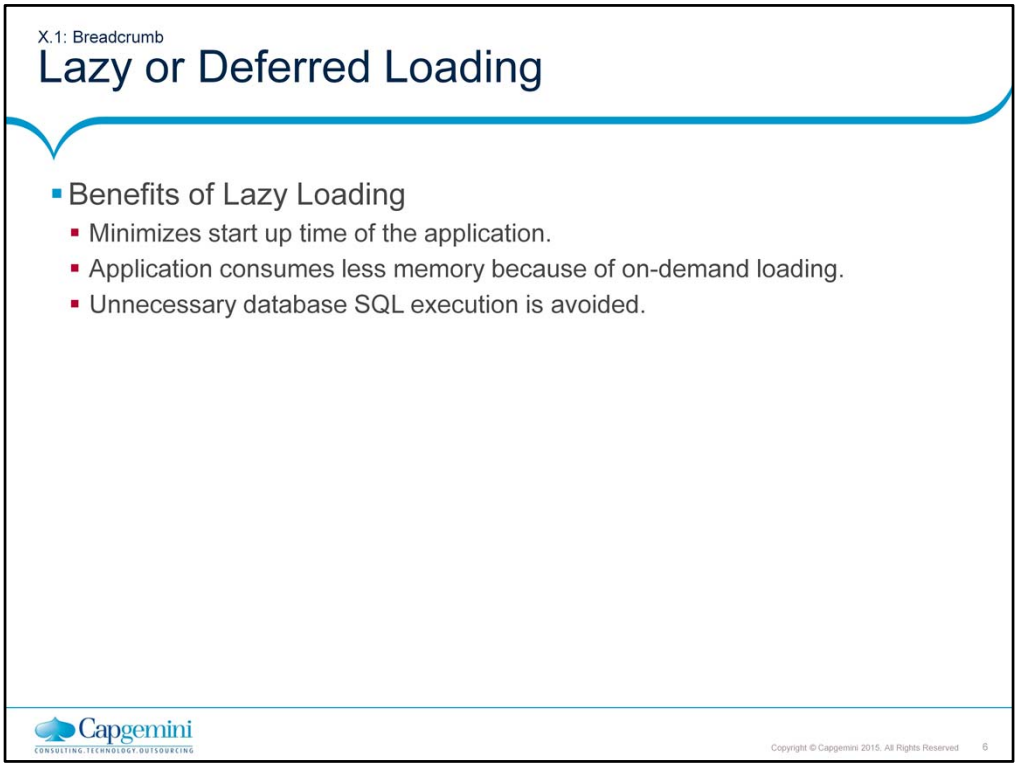
An example of this would be a customer/orders scenario. You already have the customers loaded, but you want their orders. So now you go back to the database to get orders for a particular customer. Lazy loading means that this trip to the database to get the sales information has happened
automatically.

To Implement Lazy Loadiing on POCO you have to follow the following steps

1) Set the DeferredLoadingEnabled propety to true
   Eg :- context.ContextOptions.LazyLoadingEnabled=true

2) Set the property as virtual which need to be lazy loading
   Eg:-

```
public Guid rowguid { get; set; }
public DateTime ModifiedDate { get; set; }
public virtual Contact Contact { get; set; }
public int ContactID { get; set; }
```


X.1: Breadcrumb
Lazy or Deferred Loading
Benefits of Lazy Loading
Minimizes start up time of the application.
Application consumes less memory because of on-demand loading.
Unnecessary database SQL execution is avoided.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
6

X.X: [Topic]

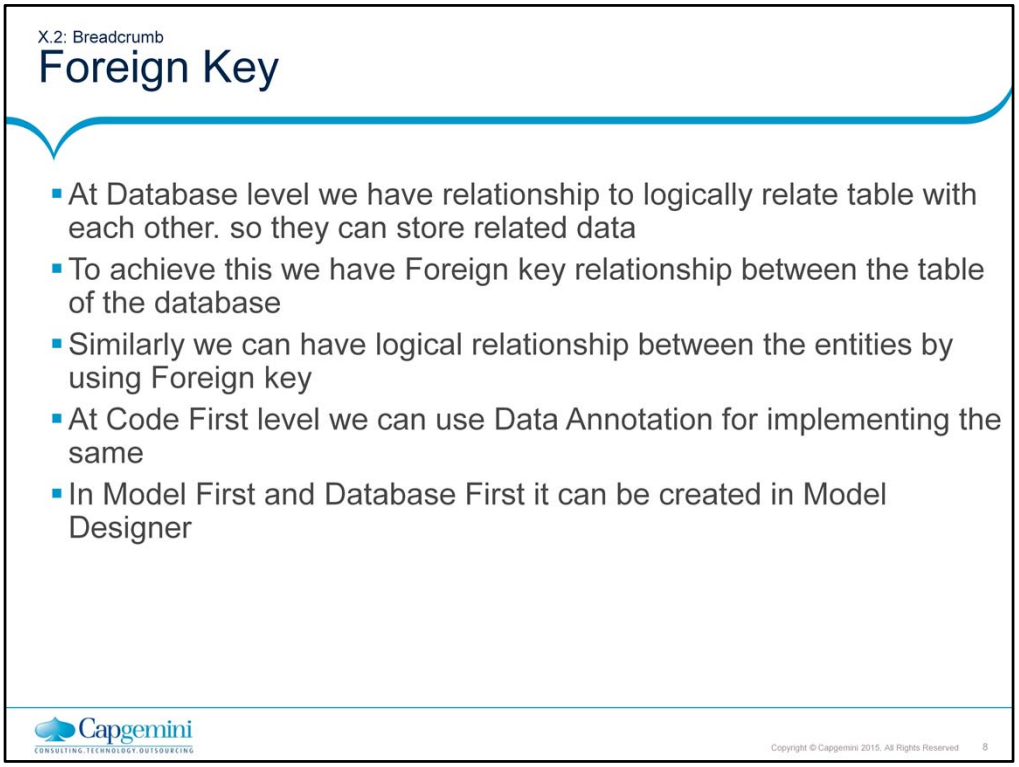
Demo

- Demo for Implementing POCO and Lazy Loading

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

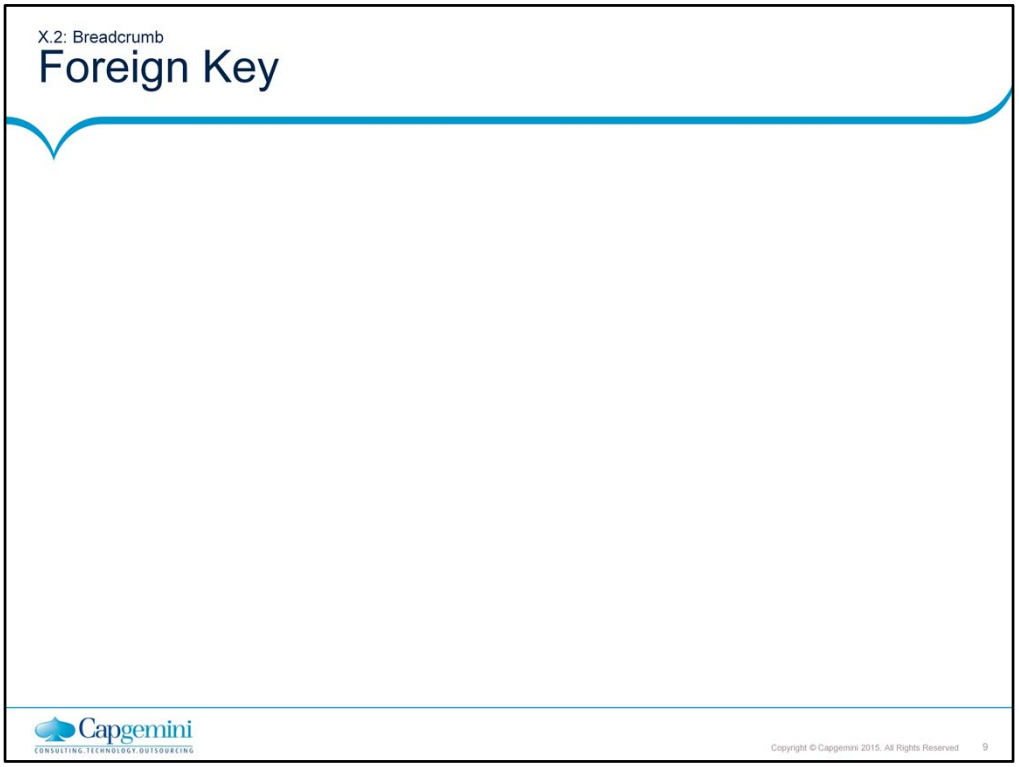
 7

Add the notes here.



Foreign Key :-

•A foreign key association, the foreign key is exposed as a property in the dependent entity.
•Exposing the foreign key allows many aspects of the association to be managed with the same code that manages the other property values.
•This is particularly helpful in disconnected scenarios
•Foreign key associations are the default in Entity Framework.
•For independent associations, the foreign keys are not exposed as properties.
•This makes the modeling at the conceptual layer somewhat cleaner because there is no noise introduced concerning the details of the association implementation.
•In the early versions of Entity Framework, only independent associations were supported.



Foreign key allow to logically relate entities in Entity Framework
Following code show an example of implementing Foreign Key relation ship in Code First Approach

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
public class Course
{
    [Key]
    public int CourseID { get; set; }
    public string CourseName { get; set; }
}

public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }
    [ForeignKey("CourseID")]
    public Course CourseID { get; set; }
}
```

X.X: [Topic]

Demo

- Demo for Implementing Foreign Keys

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

Add the notes here.

X.X: [Topic]

Lab

- Lab Topic

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

Add the notes here.

Add the notes here.

Review Question

- Which of the following features are supported by Model First Approach
  - Syntax Check
  - Lazy Loading
  - POCO
  - Foreign Keys

- Lazy Loading allow you to load you load data at runtime
  - True
  - False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Add the notes here.

Review Question

- _____ property need to set to true to enable lazy loading in the context
  - SetRuntimeLoading
  - LazyLoading
  - LazyLoadingEnable
  - LazyLoadEnable

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Add the notes here.

LINQ and Entity Framework

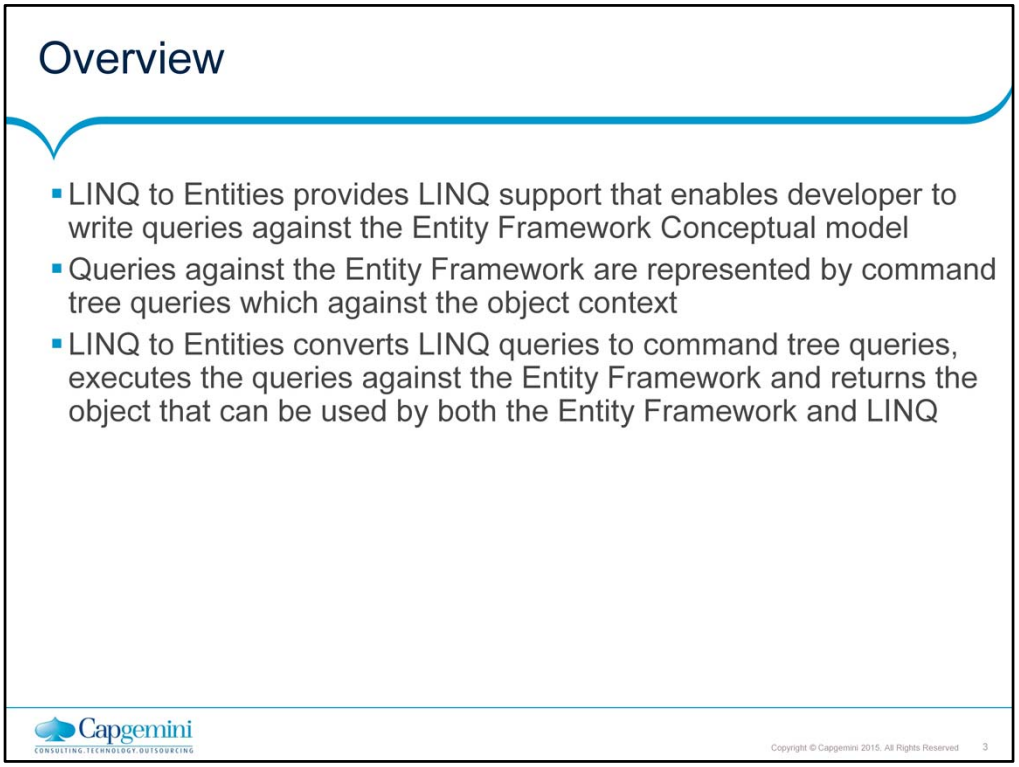
Lesson 05: Overview of LINQ to Entities

Lesson Objectives

- In this lesson we will cover the following topic
  - Basic query operations with LINQ to Entities
  - Querying and Manipulating Entity Data Model

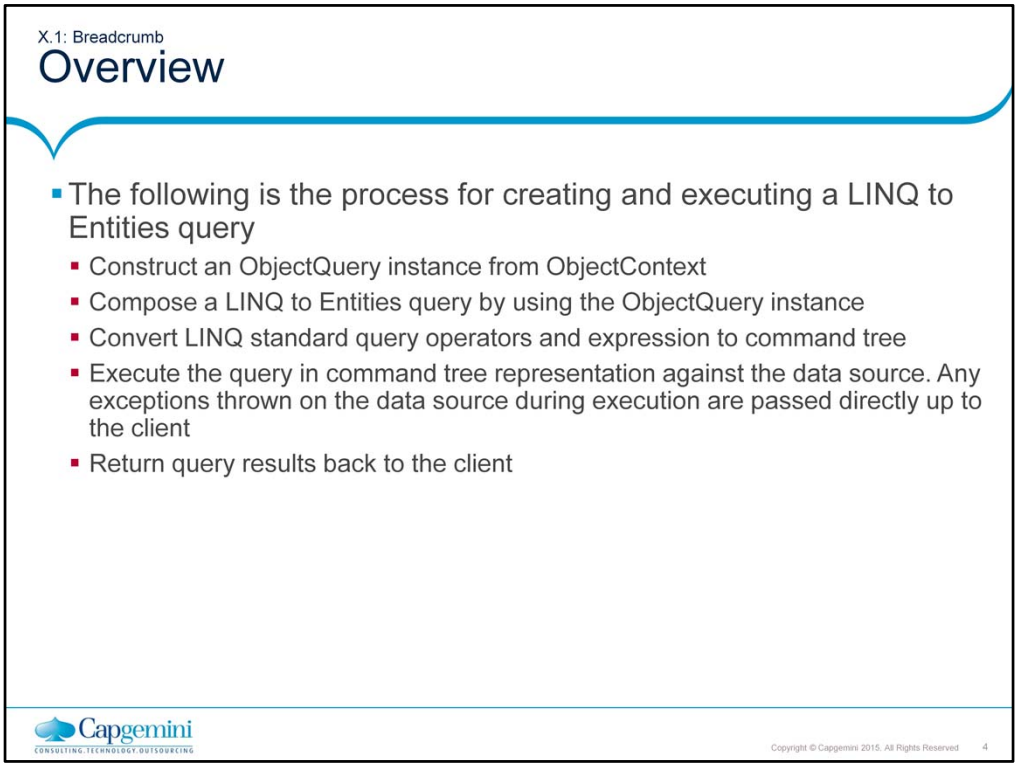
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



LINQ to Entities :-

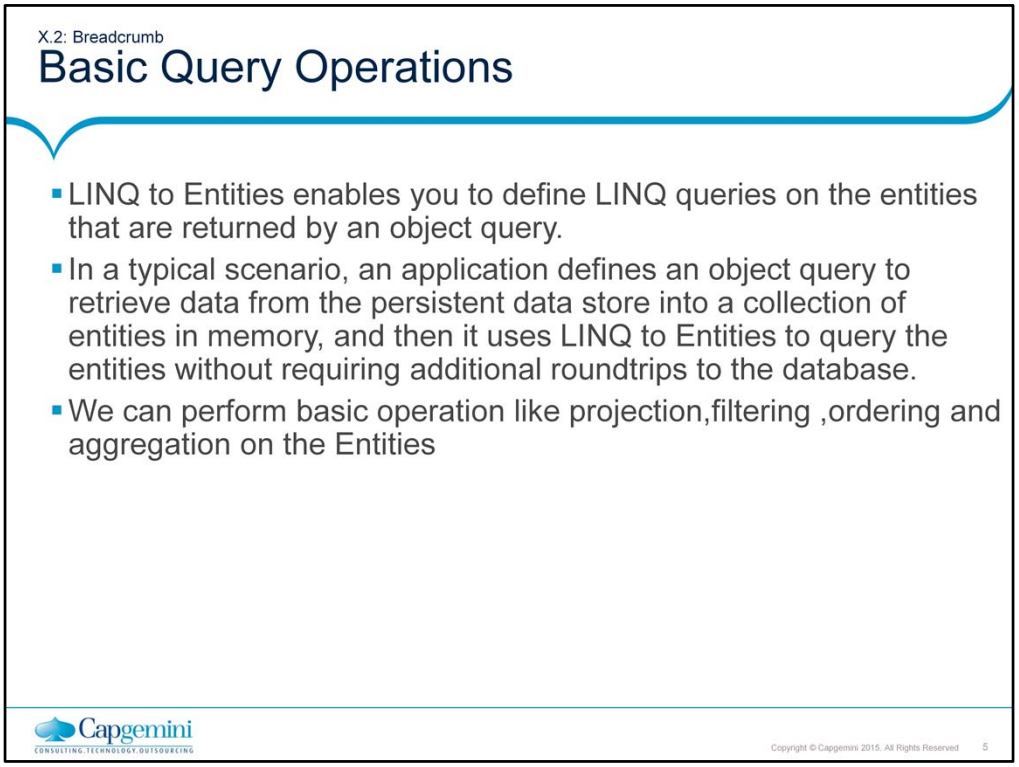
LINQ to Entities provides Language-Integrated Query (LINQ) support that enables developers to write queries against the Entity Framework conceptual model using Visual Basic or Visual C#.
Queries against the Entity Framework are represented by command tree queries, which execute against the object context.
LINQ to Entities converts Language-Integrated Queries (LINQ) queries to command tree queries, executes the queries against the Entity Framework, and returns objects that can be used by both the Entity Framework and LINQ.



ObjectContext :- Provides facilities for querying and working with entity data as objects.

ObjectQuery

IQueryable



Add the notes here.

Basic Query Operation- Projection

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    IQueryable<Product> productsQuery = from product in context.Products
                                        select product;

    Console.WriteLine("Product Names:");
    foreach (var prod in productsQuery)
    {
        Console.WriteLine(prod.Name);
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

The slide show a simple query using LINQ to Entities . It show use of select

The above query returns name of all the products from the Product table

Basic Query Operation- Filtering

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    var onlineOrders =
        from order in context.SalesOrderHeaders
        where order.OnlineOrderFlag == true
        select new
        {
            SalesOrderID = order.SalesOrderID,
            OrderDate = order.OrderDate,
            SalesOrderNumber = order.SalesOrderNumber
        };

    foreach (var onlineOrder in onlineOrders)
    {
        Console.WriteLine("Order ID: {0} Order date: {1:d} Order number: {2}",
            onlineOrder.SalesOrderID,
            onlineOrder.OrderDate,
            onlineOrder.SalesOrderNumber);
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

The slide show example of Filtering of data using LINQ to Entites. In this example we are using Where for filtering

The query will returns all the order which was placed online

Basic Query Operation- Ordering

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    IQueryable<Contact> sortedNames =
        from n in context.Contacts
        orderby n.LastName
        select n;

    Console.WriteLine("The sorted list of last names:");
    foreach (Contact n in sortedNames)
    {
        Console.WriteLine(n.LastName);
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

The above example shows use of OrderBy.

The query will return a list of contacts ordered by last name.

Basic Query Operation- Ordering

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    IQueryable<Decimal> sortedPrices =
        from p in context.Products
        orderby p.ListPrice descending
        select p.ListPrice;

    Console.WriteLine("The list price from highest to lowest:");
    foreach (Decimal price in sortedPrices)
    {
        Console.WriteLine(price);
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

The above example shows use of OrderByDescending.

The query will return sorted price list of the product from highest to lowest.

Basic Query Operation- Aggregate

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    ObjectSet<Product> products = context.Products;

    var query = from product in products
                group product by product.Style into g
                select new
                {
                    Style = g.Key,
                    AverageListPrice =
                        g.Average(product => product.ListPrice)
                };

    foreach (var product in query)
    {
        Console.WriteLine("Product style: {0} Average list price: {1}",
            product.Style, product.AverageListPrice);
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

The above example shows use of Average Aggregate function.

The query returns the average list price of the products of each style.

Basic Query Operation- Aggregate

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    ObjectSet<Contact> contacts = context.Contacts;

    //Can't find field SalesOrderContact
    var query =
        from contact in contacts
        select new
        {
            CustomerID = contact.ContactID,
            OrderCount = contact.SalesOrderHeaders.Count()
        };

    foreach (var contact in query)
    {
        Console.WriteLine("CustomerID = {0} \t OrderCount = {1}",
            contact.CustomerID,
            contact.OrderCount);
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

The above example shows use of Count Aggregate function.

The query return a list of contact IDs and how many orders each has.

Basic Query Operation- Aggregate

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    ObjectSet<SalesOrderHeader> orders = context.SalesOrderHeaders;

    var query =
        from order in orders
        group order by order.Contact.ContactID into g
        select new
        {
            Category = g.Key,
            maxTotalDue =
                g.Max(order => order.TotalDue)
        };

    foreach (var order in query)
    {
        Console.WriteLine("ContactID = {0} \t Maximum TotalDue = {1}",
            order.Category, order.maxTotalDue);
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

The above example shows use of Max Aggregate function.

The query return the largest total due for each contact ID.

Basic Query Operation- Aggregate

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    ObjectSet<SalesOrderHeader> orders = context.SalesOrderHeaders;

    var query =
        from order in orders
        group order by order.Contact.ContactID into g
        select new
        {
            Category = g.Key,
            smallestTotalDue =
                g.Min(order => order.TotalDue)
        };

    foreach (var order in query)
    {
        Console.WriteLine("ContactID = {0} \t Minimum TotalDue = {1}",
            order.Category, order.smallestTotalDue);
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

The above example shows use of Min Aggregate function.

The query returns the smallest total due for each contact ID.

Basic Query Operation- Aggregate

```
using (AdventureWorksEntities context = new AdventureWorksEntities())
{
    ObjectSet<SalesOrderHeader> orders = context.SalesOrderHeaders;

    var query =
        from order in orders
        group order by order.Contact.ContactID into g
        select new
        {
            Category = g.Key,
            TotalDue = g.Sum(order => order.TotalDue)
        };

    foreach (var order in query)
    {
        Console.WriteLine("ContactID = {0} \t TotalDue sum = {1}",
            order.Category, order.TotalDue);
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

The above example shows use of Sum Aggregate function.

The query returns the total due for each contact ID.

X.X: Basic Query Operations

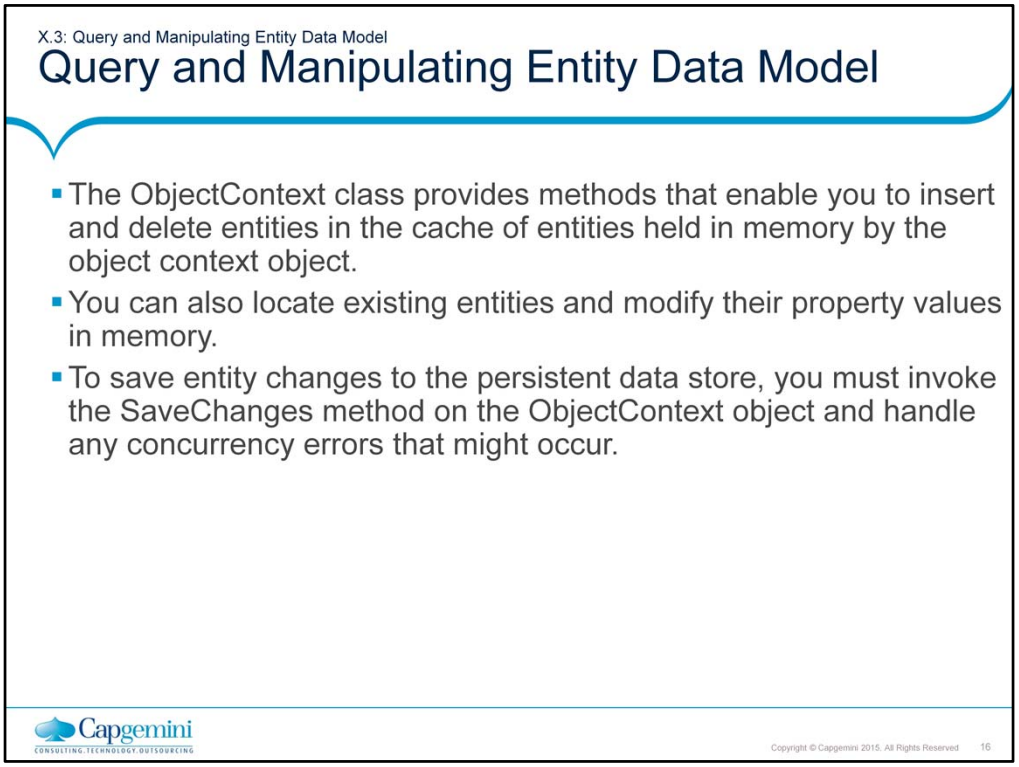
Demo

- Implementing Basic query operations in LINQ to Entities

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

Add the notes here.



Query and Manipulating Entity Data Model

To insert an entity into an object context:

1. Create an entity and set its properties
2. Invoke the `Add` method on the `ObjectContext` object

To update an existing entity in an object context:

1. Locate the entity in the object context
2. Modify property values on the entity

To delete an existing entity in an object context:

1. Locate the entity in the object context
2. Invoke the `Remove` method on the `ObjectContext` object

To save the object context changes to the data store:

1.Invoke the `SaveChanges` method on the `ObjectContext` object
2. Handle concurrency exceptions

X.X: [Topic]

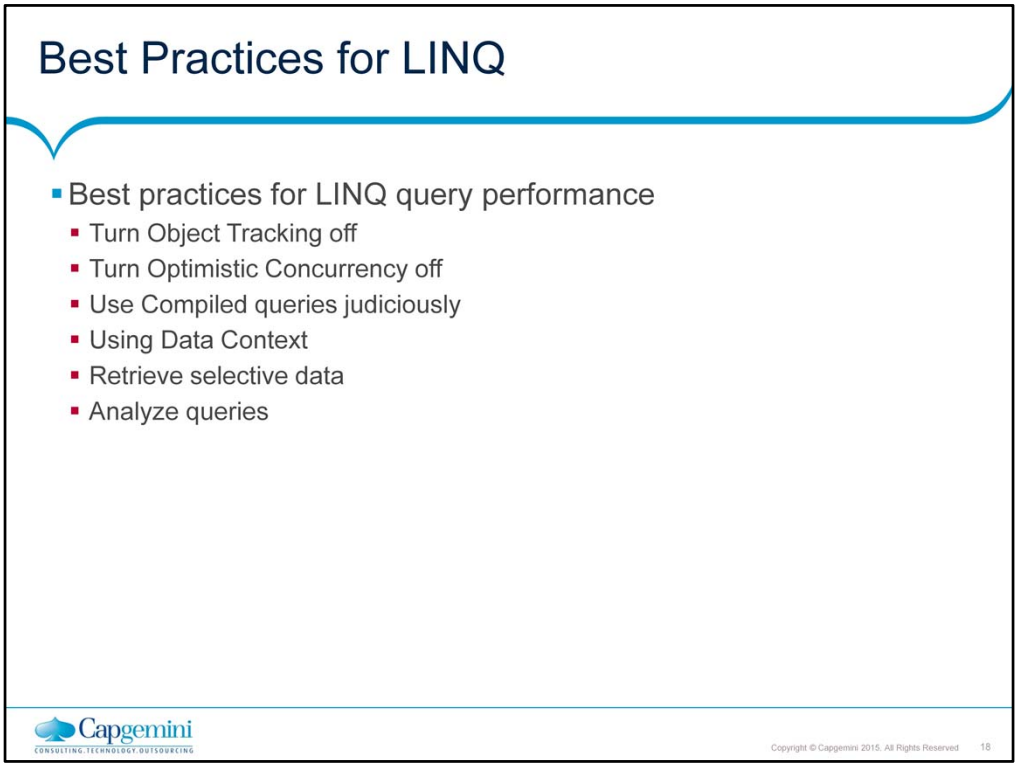
Demo

- Implementing Data Manipulation using LINQ to Entities

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

Add the notes here.



Best Practices of LINQ query performance

Turn Object Tracking off :-

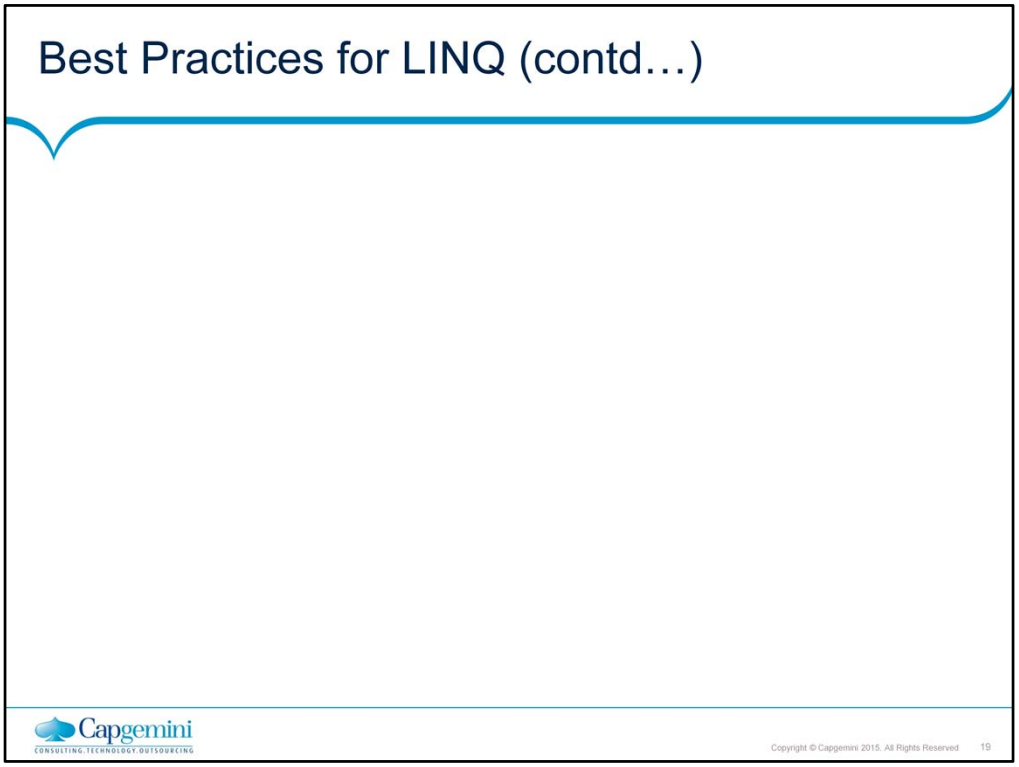
You should turn ObjectTrackingEnabled property off if you don't need it. Note that if don't need to change data but just read it, it is always advisable to turn this property off so as to turn off the unnecessary identity management of objects. The ObjectTrackingEnabled property is set to true by default. This implies that LINQ to SQL would keep track of every change that you make to your data so that it can remember those changes when you need to persist those changes to the underlying database at a later point in time. This will help boost the application's performance to a considerable extent. The following code snippet illustrates how you can turn this property off (set it to false):

```
using (IDGDataContext dataContext = new IDGDataContext())
{
dataContext.ObjectTrackingEnabled = false;
//Usual code
}
```

Turn Optimistic Concurrency off :-

Concurrency handling enables you to detect and resolve conflicts that arise out of concurrent requests to the same resource. Note that there are two types of concurrency - Optimistic and Pessimistic and LINQ follows an optimistic concurrency model by default. You should turn optimistic concurrency off unless it is needed. Note that UpdateCheck is set to Always which implies LINQ to SQL will check the property value, i.e., the value contained in that property against the value stored in the column of the database table that is mapped to that property. You should avoid using optimistic concurrency if not needed. You can use the UpdateCheck property to turn off optimistic concurrency. The following code snippet illustrates how you can set this property in the attribute of your entity class:

```
[Column(Storage="_Address", DbType="NText",
UpdateCheck=UpdateCheck.Never)]
```

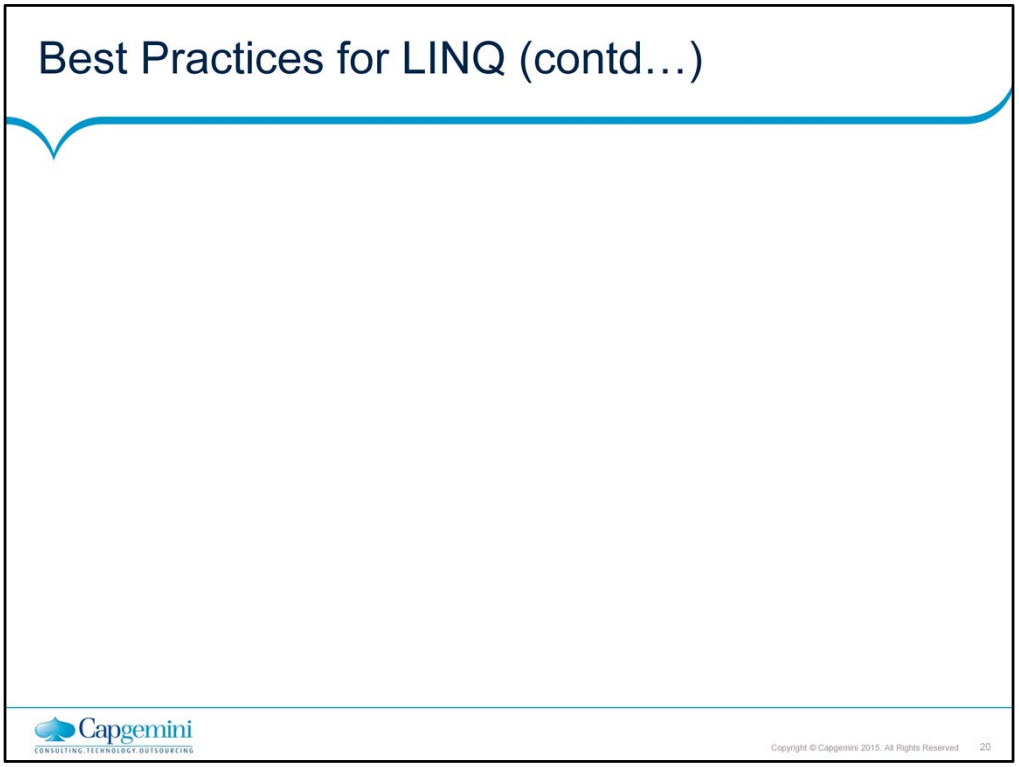


Use Compiled queries judiciously :-

You can take advantage of Compiled Query to boost query performance in your application. But, remember that compiled query could be costly when used for the first time. So, do ensure you use compiled queries only in situations where you need them, i.e., when you need a query to be used repeatedly.
At the time when a query is to be executed by the LINQ engine, LINQ to SQL translates the LINQ queries to SQL -- this is repeated every time the query is to be executed. This involves traversing the expression tree recursively again and hence it is a performance overhead. No worries at all -- you have the CompiledQuery class for the rescue
You can leverage CompiledQuery to eliminate this performance overhead for queries that need to be executed again and again. A word of caution though: Use CompiledQuery judiciously and only when it is needed.

Using Data Context :-

You should not dump all the database objects into one single DataContext. The DataContext should represent one unit of work. This approach if followed would reduce the identity management and object tracking overhead involved. Also, you should only attach the objects to your data context those have been changed since the time they were read into the memory.



Retrieve selective data :-

You should retrieve only the data that is needed and avoid retrieving all of the data. To achieve this, you can take advantage of the Take and Skip methods. You can also filter the data to be retrieved using DataLoadOptions.AssociateWith so that only the required data is returned. The following code snippet illustrates how DataLoadOptions.AssociateWith can be used.

```
using (IDGDataContext dataContext = new IDGDataContext())
{
DataLoadOptions dataLoadOptions = new DataLoadOptions();
dataLoadOptions.AssociateWith<Customer>(customer =>
customer.Address.Where<Address>(address => address.PinCode == 500016));
dataContext.LoadOptions = dataLoadOptions;
}
```

Analyze queries :-

You should always analyze your LINQ queries and take a monitor the generated SQL. You can set the Log property of the data context to see the generated SQL.

```
using (IDGDataContext dataContext = new IDGDataContext())
{
dataContext.Log = Console.Out;
}
```

This will help you to understand how your LINQ query has been translated to SQL and any additional columns or extra data that is retrieved when the SQL query is eventually executed.

X.X: [Topic]

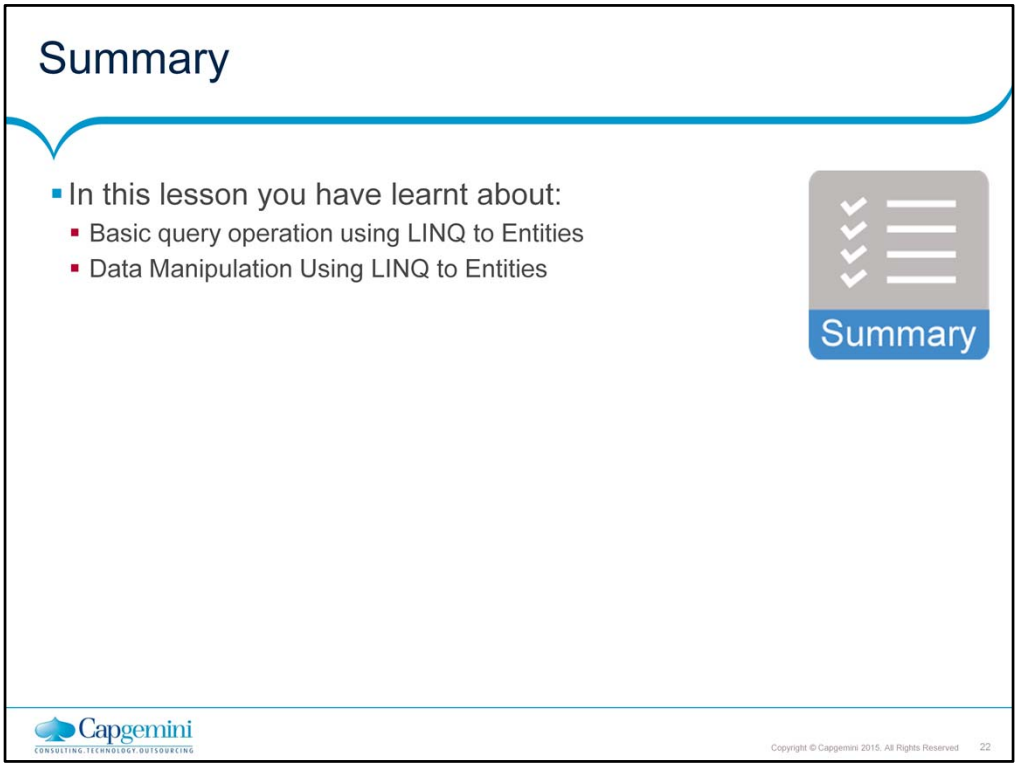
Lab

- Lab Topic

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

Add the notes here.



Add the notes here.

Review Question

- Which of the following allows to access Entities using declarative query syntax?
  - LINQ to Array
  - LINQ to SQL
  - LINQ to DataSet
  - LINQ to Entities

- ObjectContext Represents a typed query against a conceptual model in a given object context.
  - True
  - False
- Question 3: Fill in the Blanks

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

Add the notes here.

Review Question

- _____ method allow to save change to database
  - SafeChanges()
  - SaveChanges()
  - AcceptChnage()
  - ImplementChnages()

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

Add the notes here.

LINQ and Entity Framework
Lab Book

Table of Contents

Getting Started

Overview

This lab book is a guided tour for learning HTML version x.x. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given.

Setup Checklist for HTML

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- Intel Core i3 or higher
- Microsoft Windows 7.
- Memory: 1GB of RAM (2GB or more recommended)
- Internet Explorer 10.0 or higher
- Google Chrome
- Connectivity to Sql Server
- LocalDB

Please ensure that the following is done:

- Visual Studio 2012 or above.
- .Net Framework 4.5 or above.

Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory html_assgn. For each lab exercise create a directory as lab <lab number>.
- Download all files required to complete assignments from: http://pace.patni.com/TechRS/download.asp?course=Internet_HTML
- You may also look up the on-line help provided in the MSDN library.

Learning More (Bibliography if applicable)

- http://msdn.microsoft.com
- http://www.asp.net/entity-framework
- https://msdn.microsoft.com/en-in/data/ef.aspx
- Entity Framework 6 Recepies by Apress publication

Problem Statement/ Case Study (If applicable)

Give the case study used for this lab book here. If applicable.

Lab 1. LINQ Basics

Goals	Understand the process of Implementing LINQ to a Collection Learn to use LINQ Learn to use LINQ Operators
Time	60 minutes

1) Create a console application and add class named Employee with following field.

Employee Class

EmployeeID (Integer)
FirstName (String)
LastName (String)
Title (String)
DOB (Date)
DOJ (Date)
City (String)

2) Create a Generic List Collection empList and populate it with the following records.

EmployeeID	FirstName	LastName	Title	DOB	DOJ	City
1001	Malcolm	Daruwalla	Manager	16/11/1984	8/6/2011	Mumbai
1002	Asdin	Dhalla	AsstManager	20/08/1984	7/7/2012	Mumbai
1003	Madhavi	Oza	Consultant	14/11/1987	12/4/2015	Pune
1004	Saba	Shaikh	SE	3/6/1990	2/2/2016	Pune
1005	Nazia	Shaikh	SE	8/3/1991	2/2/2016	Mumbai
1006	Amit	Pathak	Consultant	7/11/1989	8/8/2014	Chennai
1007	Vijay	Natrajan	Consultant	2/12/1989	1/6/2015	Mumbai
1008	Rahul	Dubey	Associate	11/11/1993	6/11/2014	Chennai
1009	Suresh	Mistry	Associate	12/8/1992	3/12/2014	Chennai
1010	Sumit	Shah	Manager	12/4/1991	2/1/2016	Pune

3) Now once the collection created write down and execute the LINQ queries for collection as follows

i) Display detail of all the employee
ii) Display details of all the employee whose location is not Mumbai
iii) Display details of all the employee whose title is AsstManager
iv) Display details of all the employee whose Last Name start with S
v) Display a list of all the employee who have joined before 1/1/2015

vi) Display a list of all the employee whose date of birth is after 1/1/1990
vii) Display a list of all the employee whose designation is Consultant and Associate
viii) Display total number of employees
ix) Display total number of employees belonging to “Chennai”
x) Display highest employee id from the list
xi) Display total number of employee who have joined after 1/1/2015
xii) Display total number of employee whose designation is not “Associate”
xiii) Display total number of employee based on City
xiv) Display total number of employee based on city and title
xv) Display total number of employee who is youngest in the list

Lab 2. Creating Entity Data Model

Goals	**At the end of this lab session you will be able to:** • **Understand the need for frames in web pages.** • **Create and work with frames.** • **Manage large content with frame**
Time	**30 Minutes**

Part 1:-

Using Code-First approach

Solution:-
Create a Console application and name the application as **CodeFirstConsoleApp**

After the project is created Now we have to add the Entity framework Library to Project .For that we will Nuget Package Manager Dialog

To Open Nuget Package Manager Dialog we need to follow the following step

Tools →Nuget Package Manager →Manage Nuget Package for the Solution

In the dialog box for Nuget Package Manager select entity framework and click on install this will install EntityFramework to the project
After installing the EntityFramework to the project we can see that EntityFrame dll file has been added to the References folder in Solution Explorer
Now as we have added the EntityFramework dll to the project now we have to add the Entity and Context to the project.

To add a entity class In the Solution Explorer right click on the project name than Add → Class and name the class as Product.cs
Once the class has been created add the following code to the class

After adding the class now we have to add a Context Class to the project .Context class will allow to perform database operation like add ,delete etc.
Context Class will always inherit from DbContext class available in System.Data.Entity namespace
After adding the context class add the following code to the class
In the Program.cs class file add the following code
When the above code is executed it will create the database and table based on the Entity and the above record into the table
After executing the application we will get the following output
Now we will check database which is created for that open Sql Server Management Studio and connect to the default instance. In the object explorer you can see the database and table being created

Part 2:-
Using Database First Approach

Solution:-
Open Sql Server Management Studio and Create a database name MusicStore and add aTable named Album with the following fields

Album
AlbumID
Name
Genre
Year
Price

Add some dummy record into the table.

Now create a console application name DatabaseFirstConsoleApp and add the entityframework as done in the previous example

Once the project in create and entity framework library is added . Now we have a Entity data Model to the project .

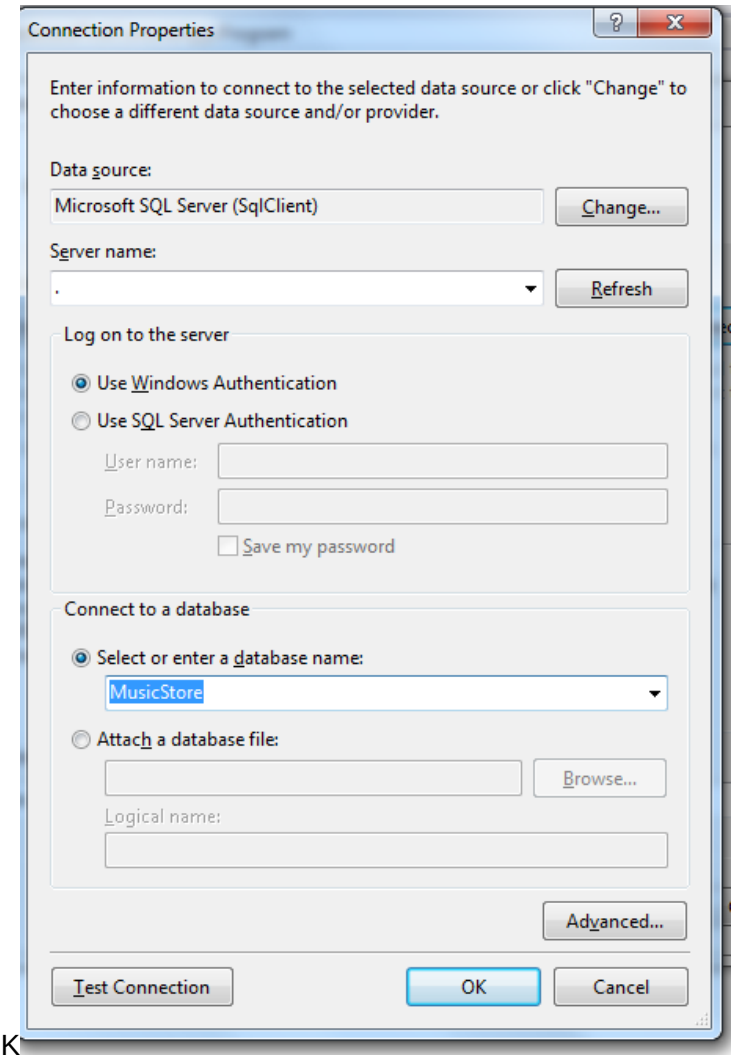
To add a Entity data Model to the Project in the solution explorer right click in the project Add→ New Item. Under the New Item dialog box select Ado.Net Entity Data Model and give name as MusicStoreModel and click on Add

As we add the Entity Data model to the project . Entity Data model wizard popup in which we have different option for initializing the entity data model

In that dialog box **select EF Designer from database** and click on Next

Now on the next window we have select database for Model creation so now click on New Connection button in Choose Your Data Connection

In the Connection properties dialog box provide the Database server name and select the database you want to use. Click on Test Connection to test the connection and then click on



OK

Once the connection test is passed then click on OK

Now we can see the new connection string which we have created and a option to save the connection string in App.config or web.config file

Now Click on Next and Choose your Database Object and Settings option will be prompted .

In that we have select all the database object which we need to add to our Model.

We have select the Album table as we have only one table in the database

Now click on Finish this will add the EDM to the project and create all the required code.

In the above image we can see the model name MusicStoreModel.edmx containing Album Entity and the highlighted region show files generated for MusicStoreModel.edmx

Now as the model is created we can write code to interact with database and perform read/write operations.

To display details of the Albums write down the following code in Program.cs File

Output:-

To add an Album into the database table write the following code in Program.cs File

Output:-

Lab 3. Creating Entity Data Model Using Model First Approach.

Goals	Understand the process of Creating Entity Data Model using Model First Approach and Creating Complex Type Learn to use of Model First Approach to create database tables
Time	60 Minutes

Model First Approach :-

Model First allows you to create a new model using the Entity Framework Designer and then generate a database schema from the model. The model is stored in an EDMX file and can be viewed and edited in the Entity Framework Designer.

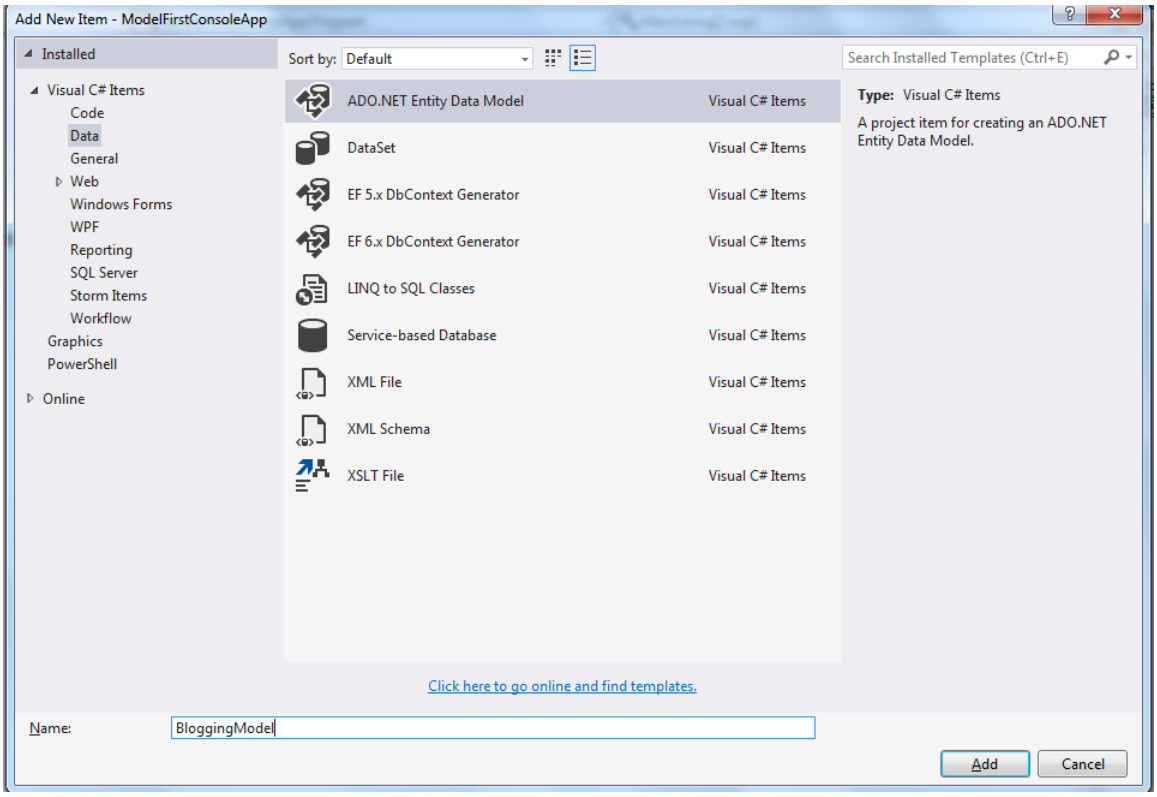
Solution:-

Open Visual Studio and create a console application named ModelFirstConsoleApp. After creating the project add Entityframework to the project using Nuget Package Manager

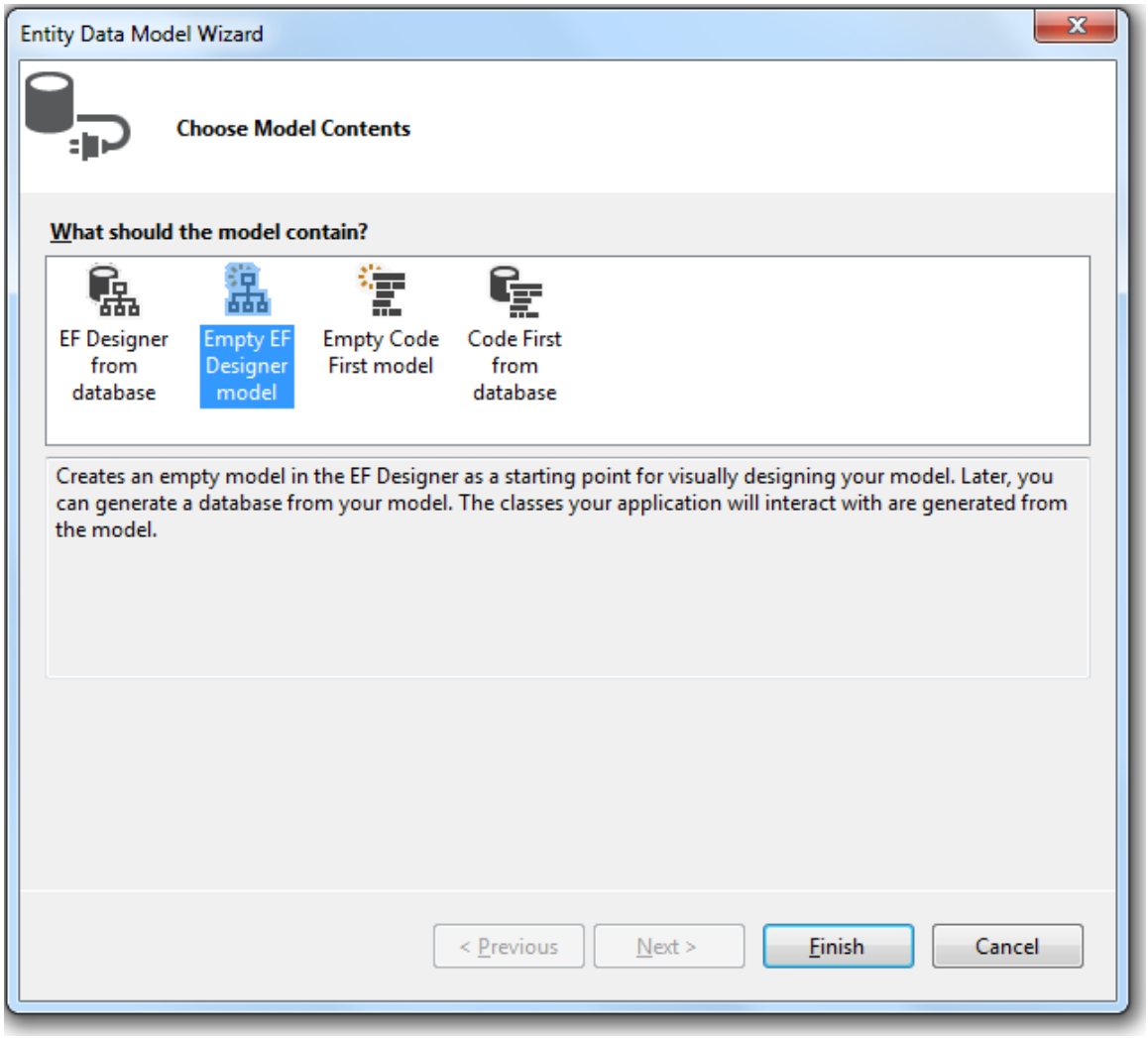
After creating the project and adding Entityframework to the project .Now we have a add a model to the project..

For adding a model to the project Right Click on the Project in Solution Explorer →Add→New Item

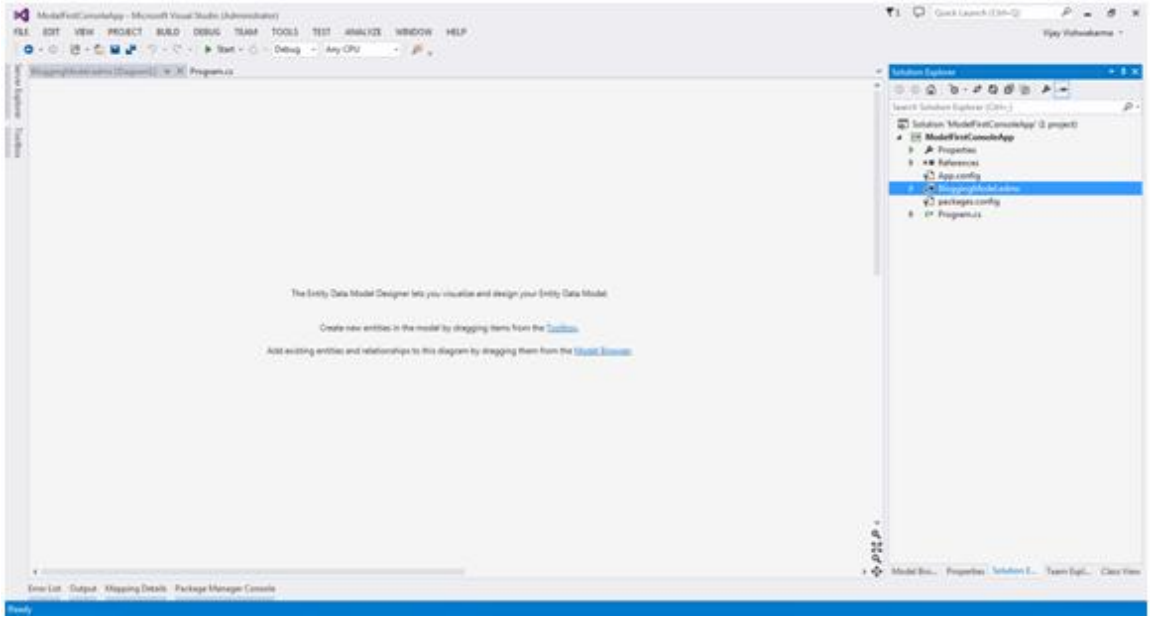
Now in the add new item dialog box select ADO.Net Entity Data Model and named it as BloggingModel and click on ADD



In the Entity Data Model Wizard select Empty EF Designer Model and click on Finish .

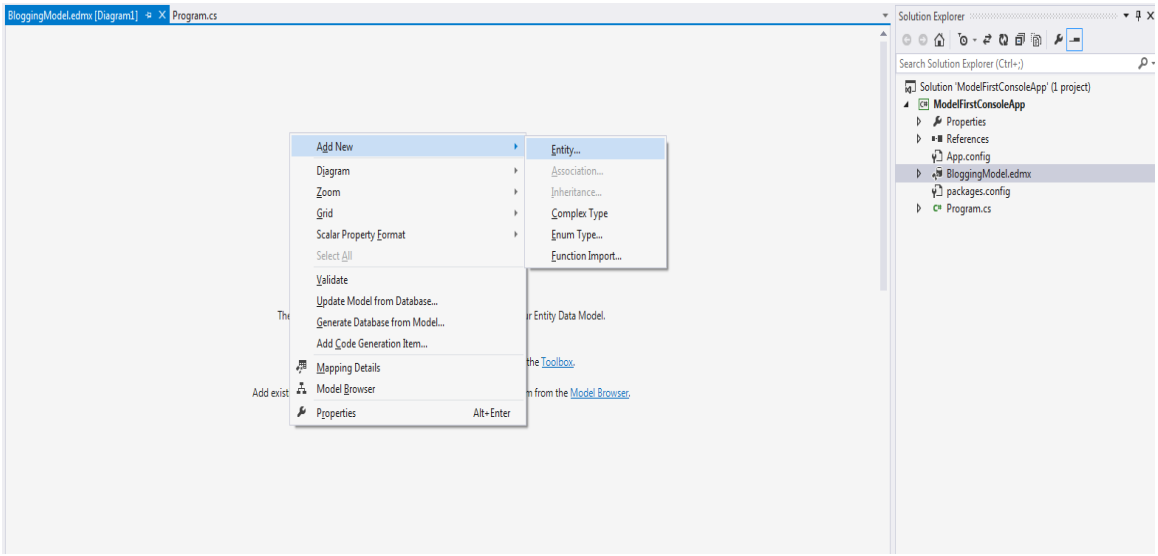


This will open up an empty EDMX on which we create Model from Scratch



Now we have to add entity in this model and create a database form it.

To add a Entity , Right click on the open edmx file and select Add New → Entity



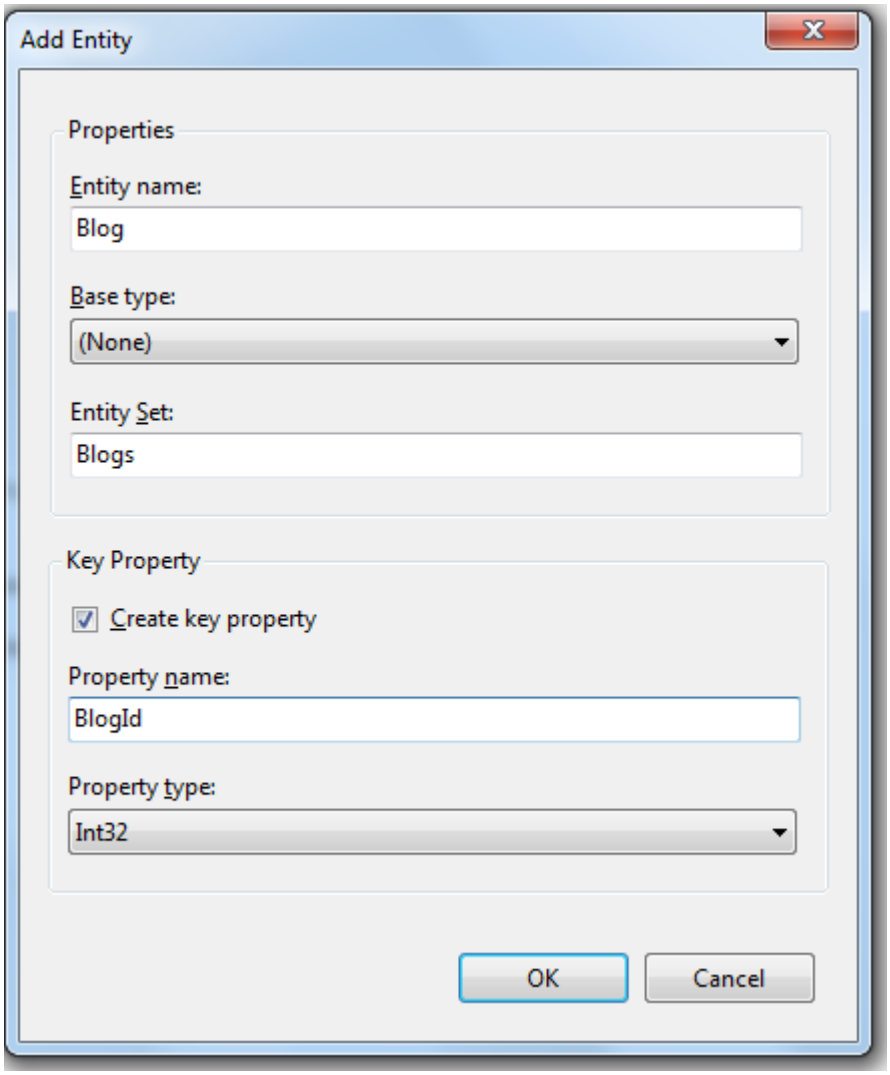
After you select you will be prompted with the following Dialog for creating a new entity

The Dialog Display information like

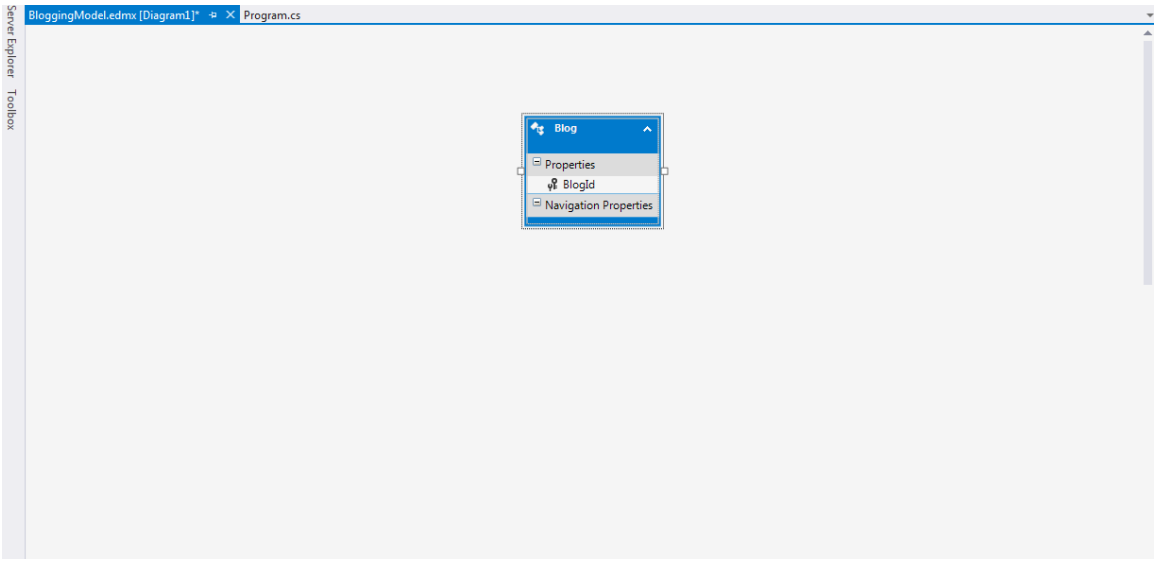
Entity Name :- This will the name of Entity
Base Type :- Shows the base type of the entity
EntitySet :- Show the name of the Entity Set

It also show the option for creating Key Property where we have to specify Key Property Name and its Data Type

For our example we will Name the entity as Blog and Entity Set as Blogs .Entity Set is automatically Pluralize as we add the name of the Entity. and change Key Property from Id to BlogId

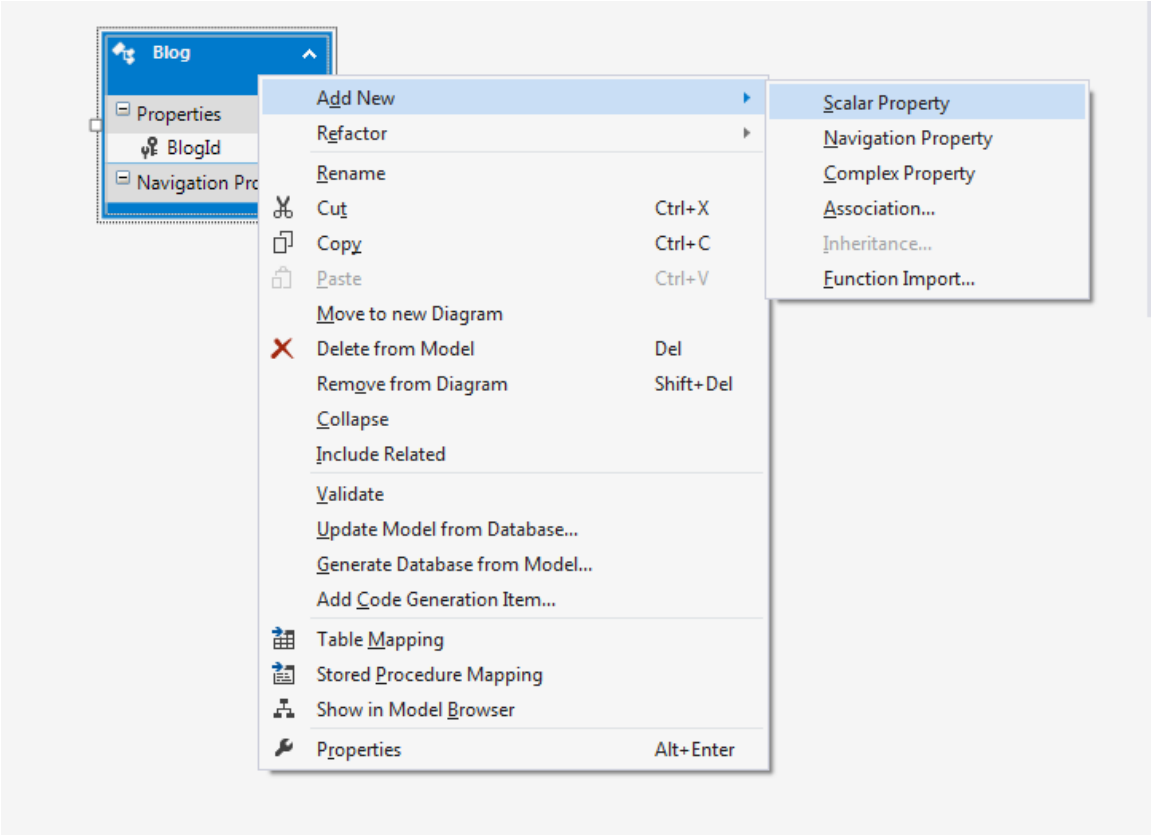


After changing all the details clik on OK .This will add the entity to EDMX designer

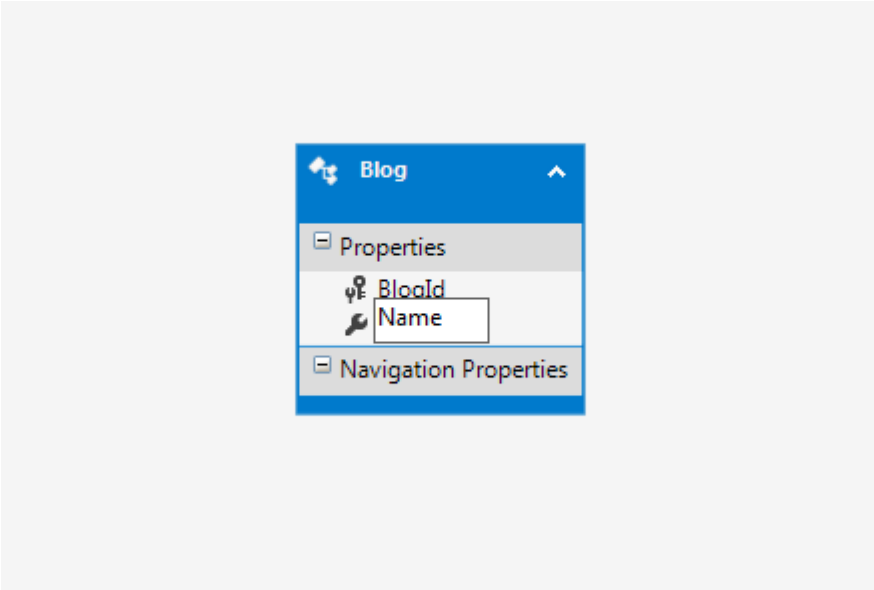


Now we have add Field to this entity for storing other information of Blog.

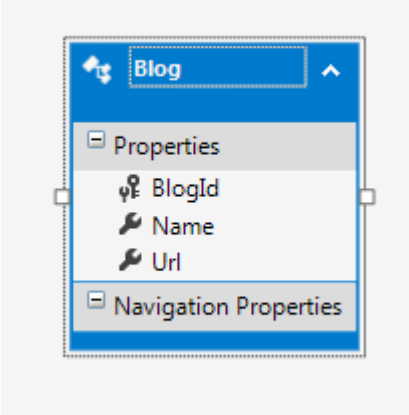
To add a Field or property to the Entity right on the Entity and select Add New → Scalar Property



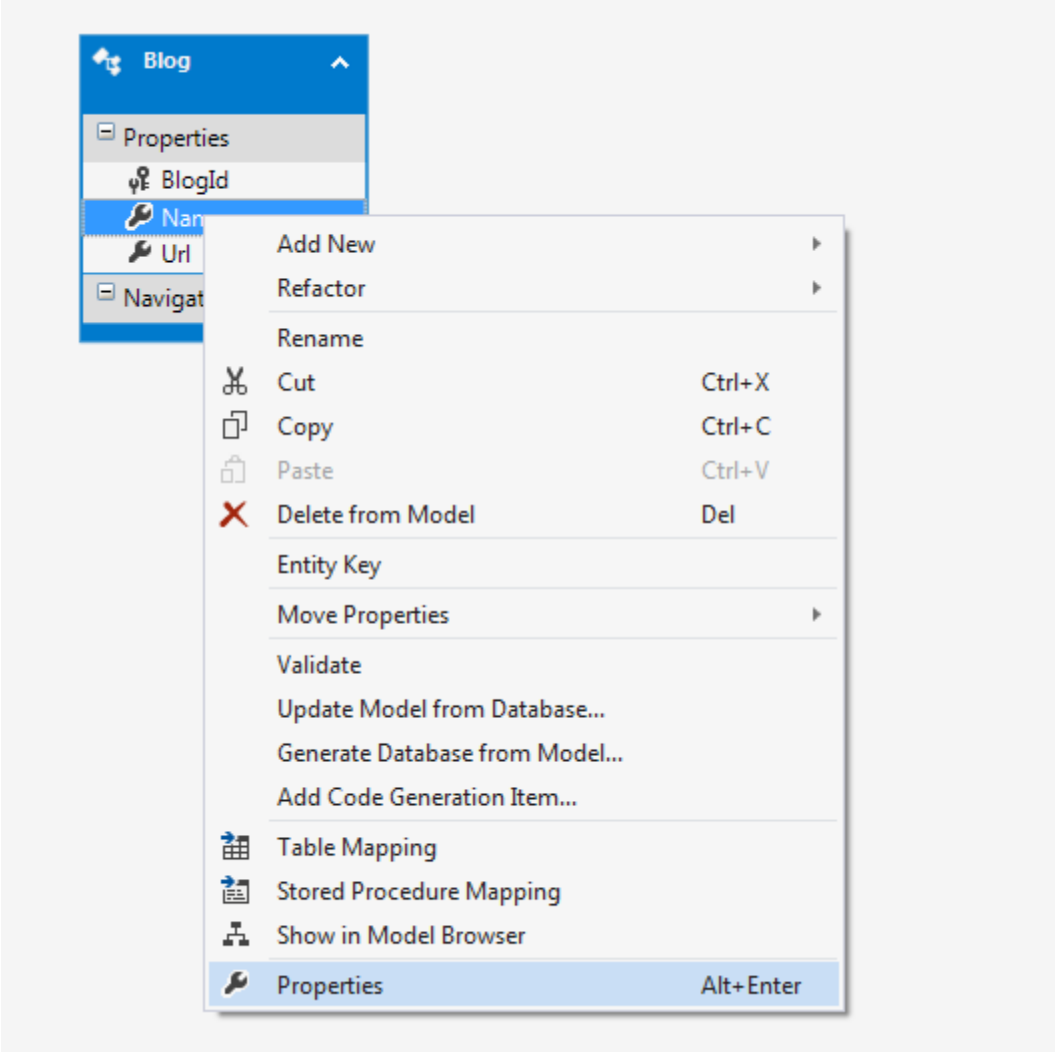
After selecting the Scalar Property , name it as Name

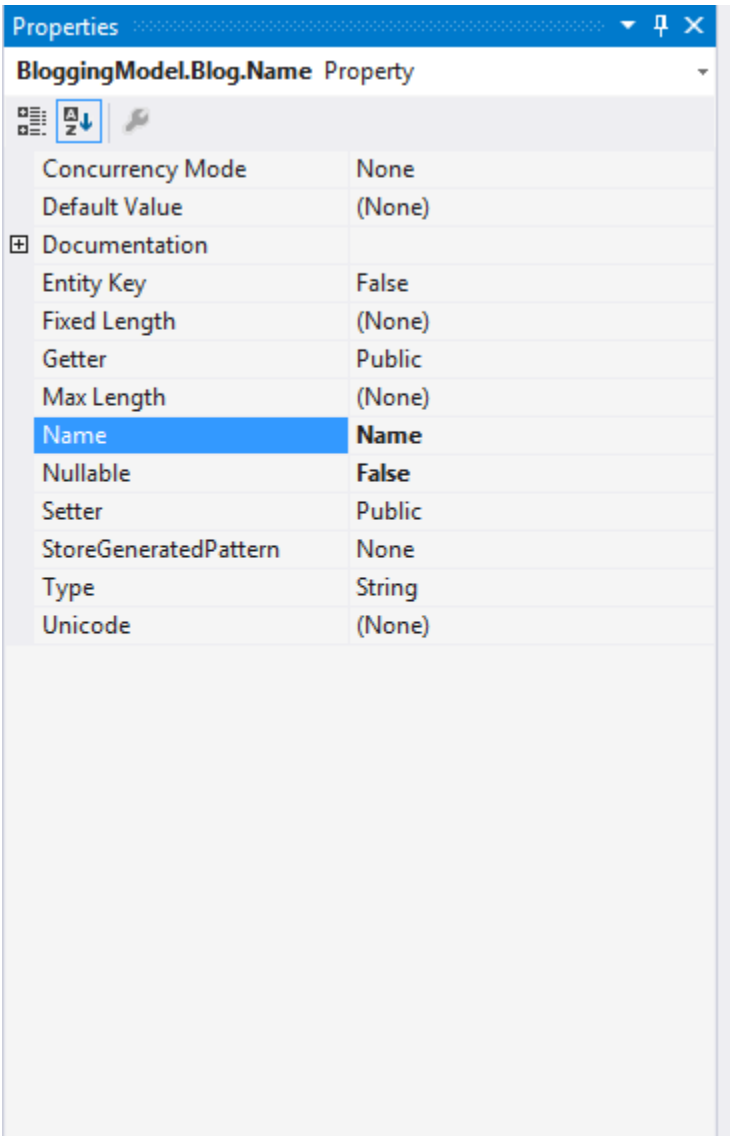


Similarly add another property named URL to the Entity. After adding the URL property the Entity Will look as follows



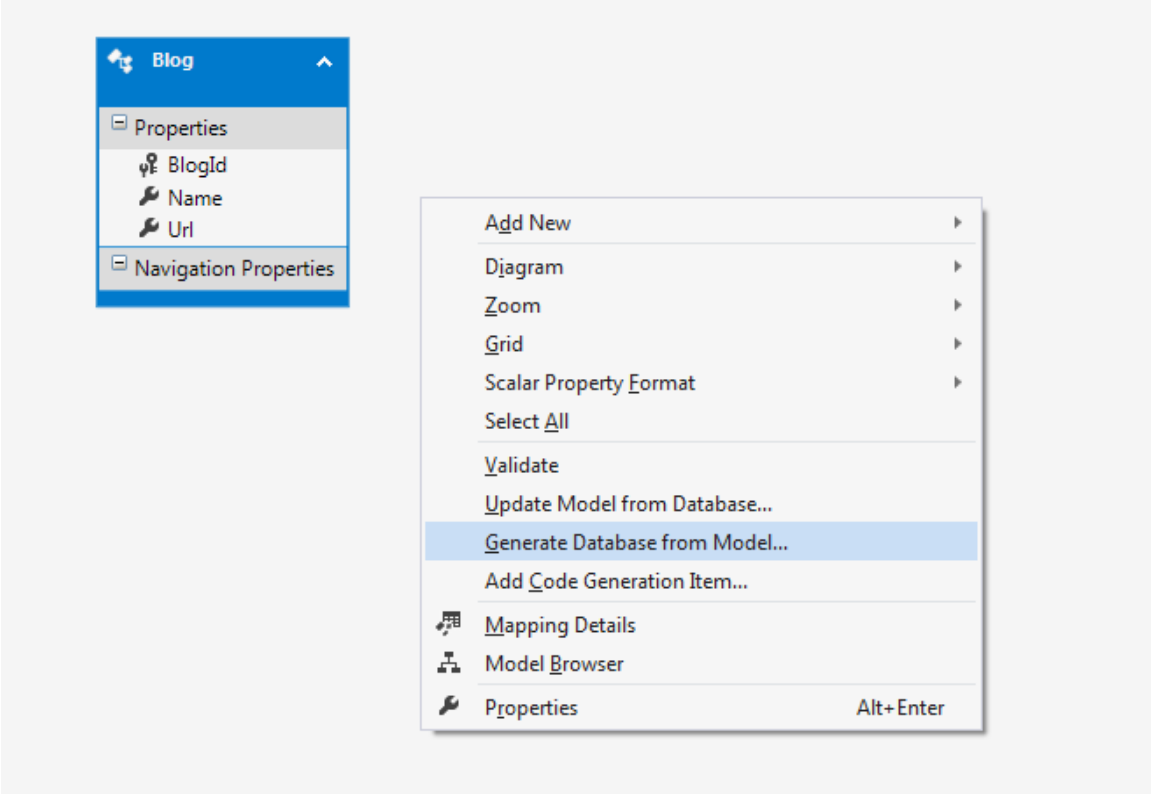
We can view the properties of each Property add the Entity by just right clicking and selecting the properties option





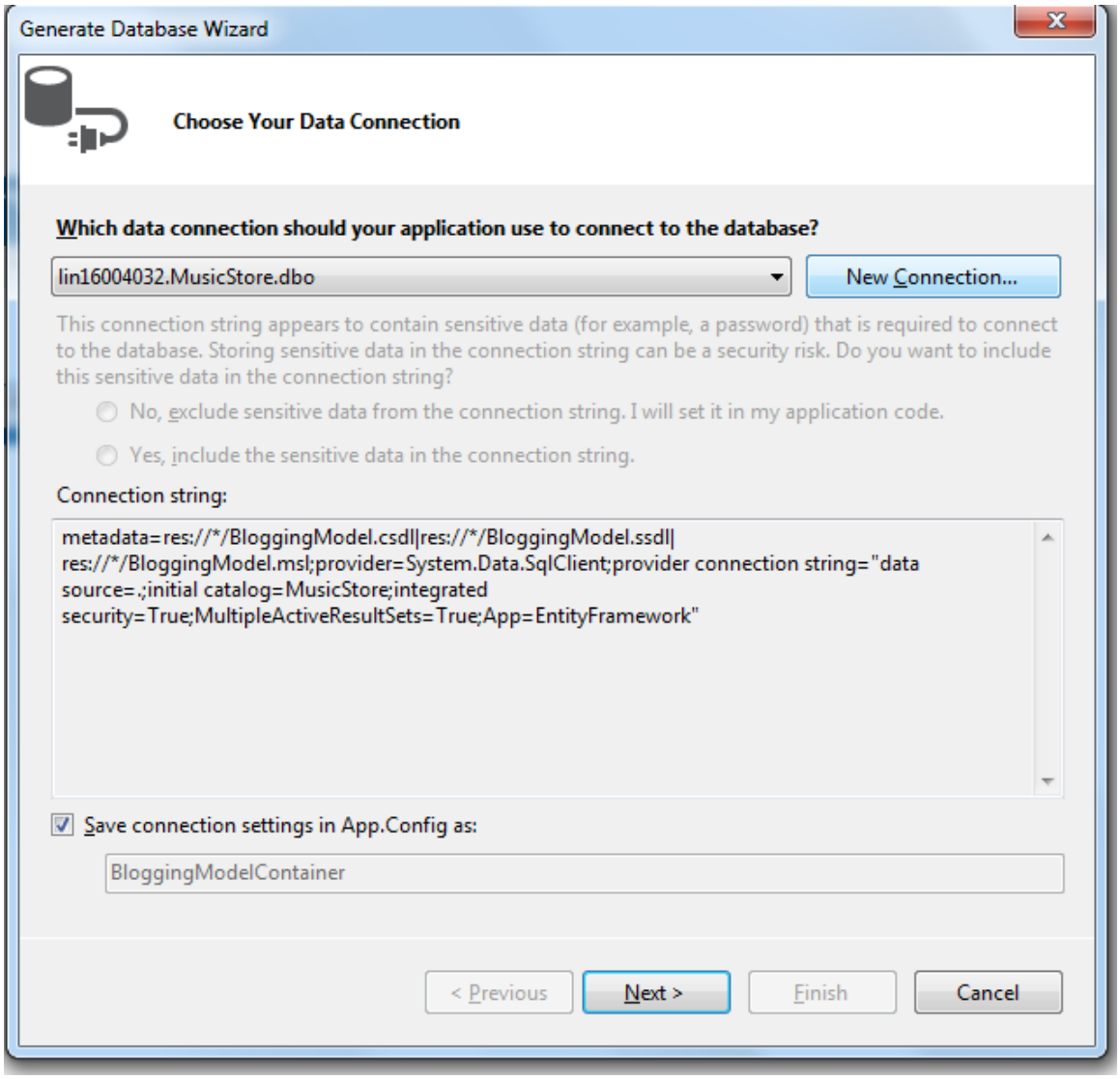
We can see above the properties show a lot of detail about the Name scalar property add to the entity .

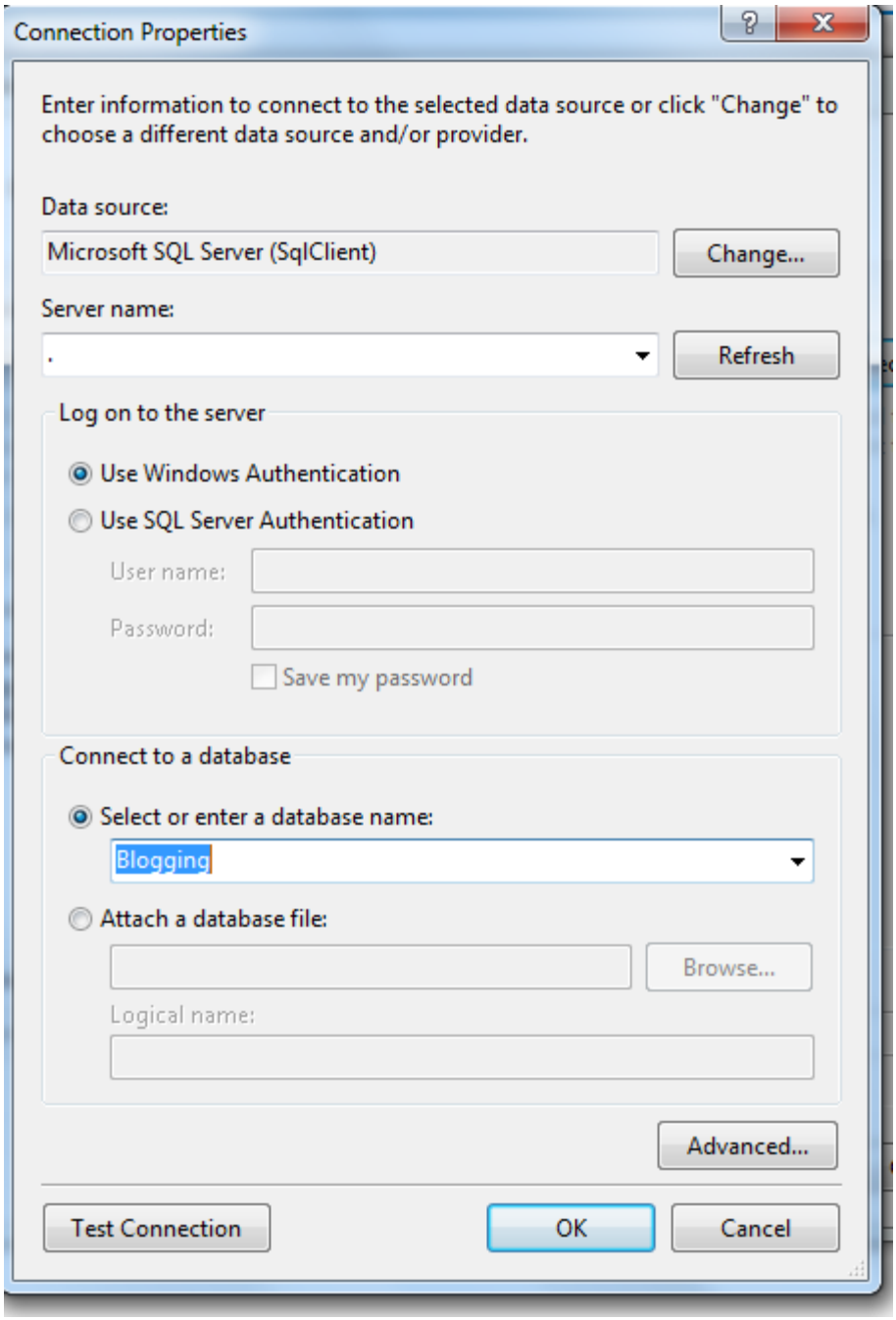
Once the Entity has been created , now we have to create Database from the Model . To create the database from the model right click on the EDMX designer and select Generate Database from Model

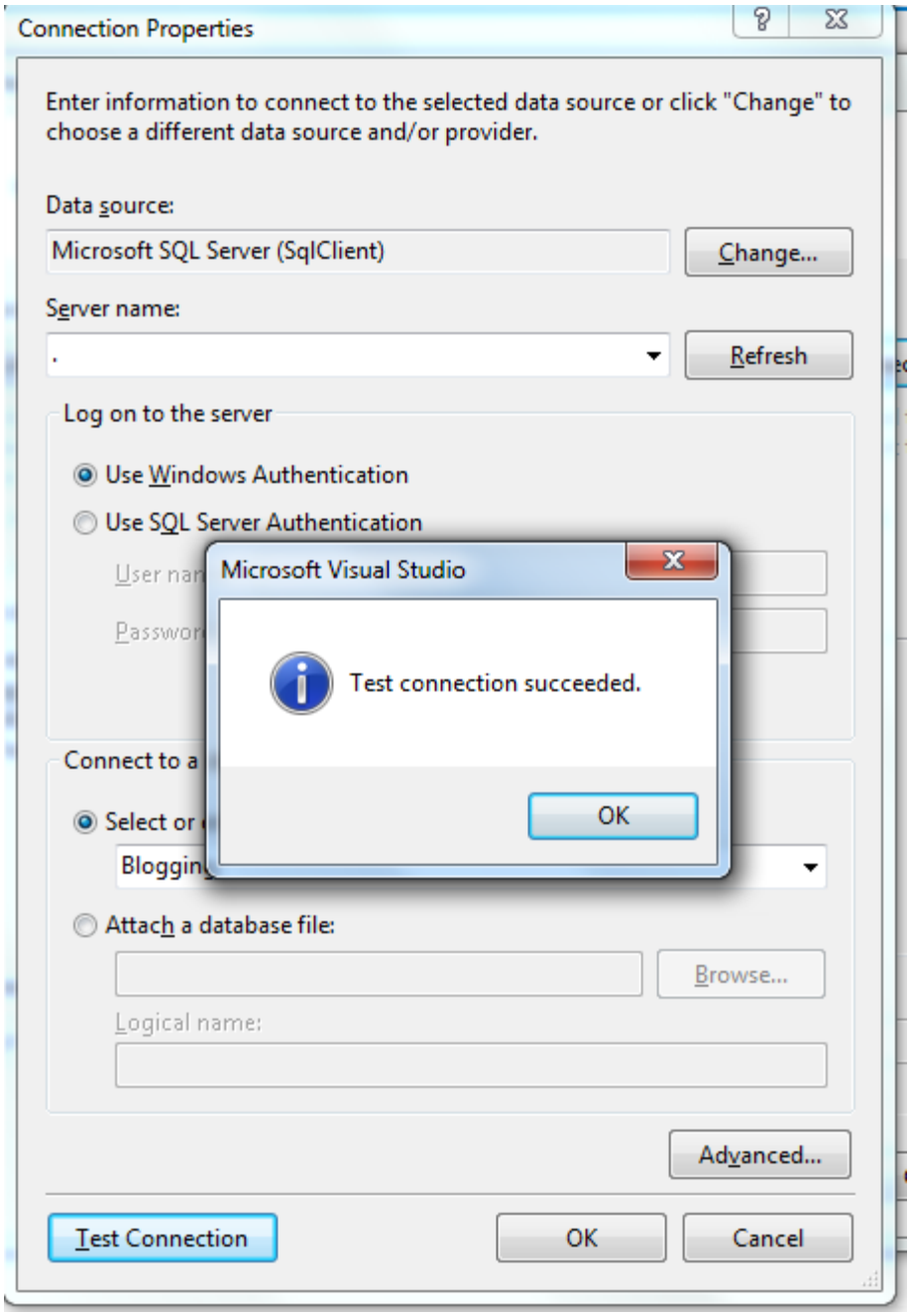


After you click on the Generate Database from Model , Generate Database Wizard will be prompted .

In the Generate Database Wizard click on the New Connection to create a new connection .

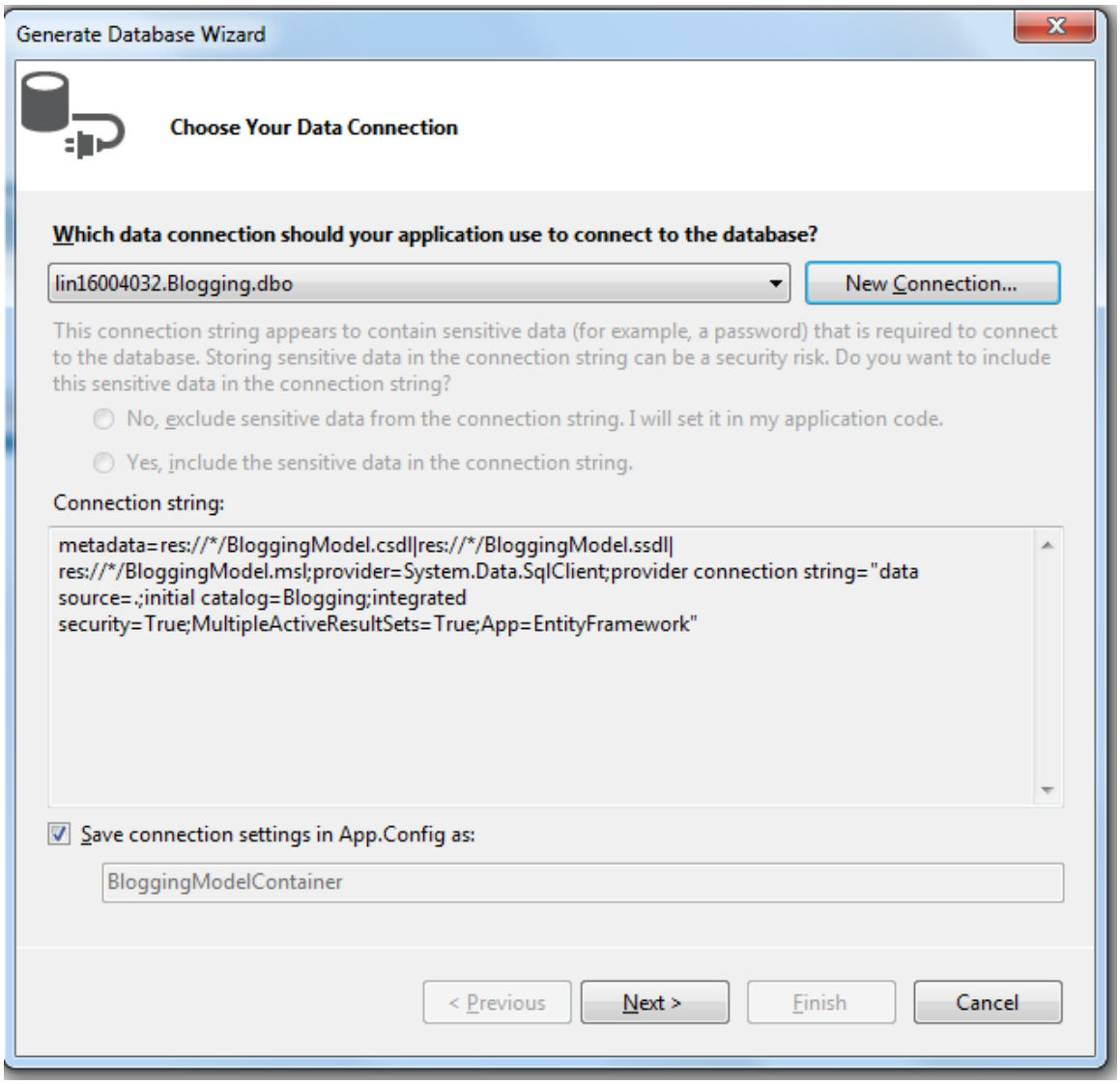




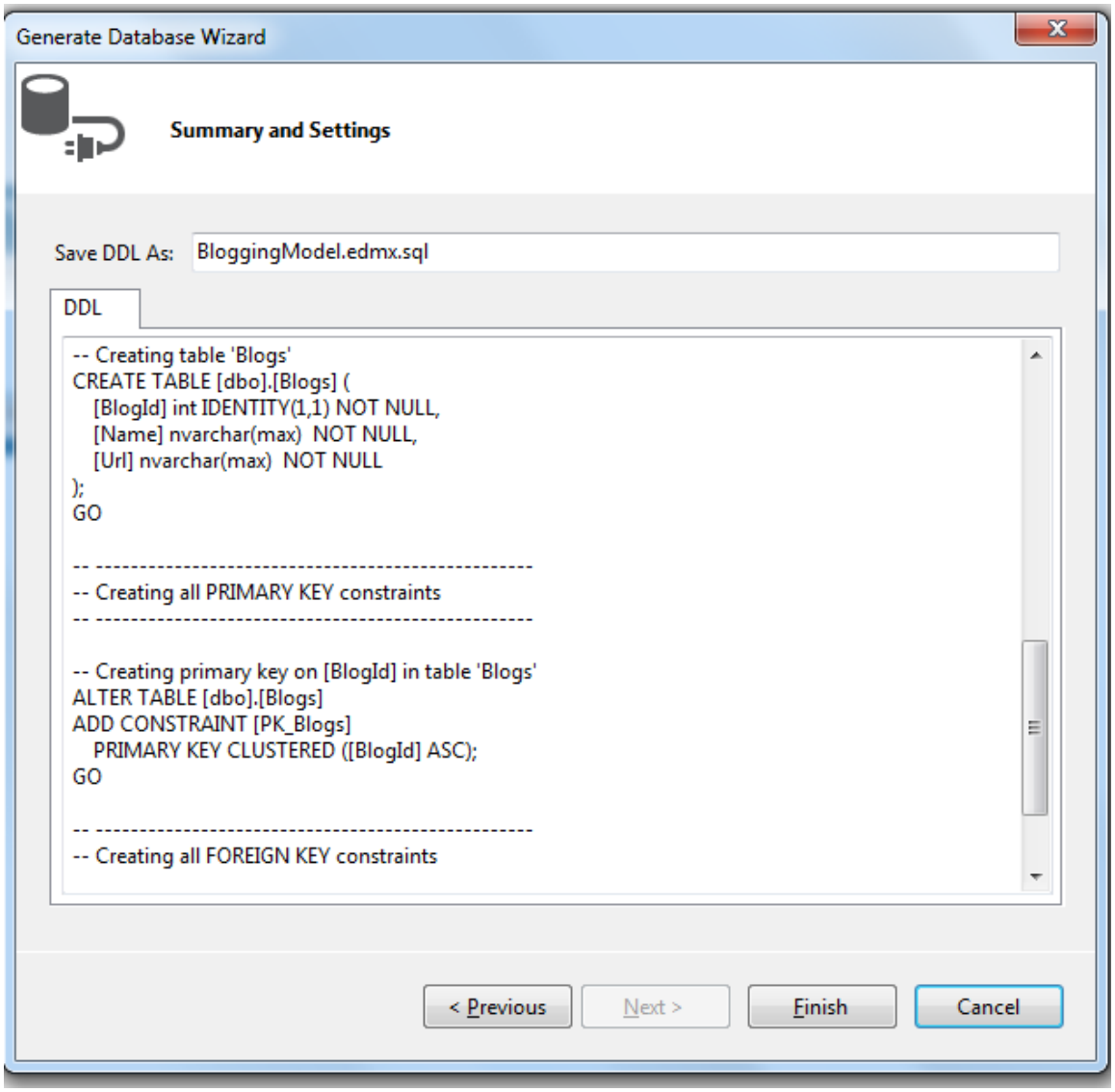


After testing the connection click on OK .

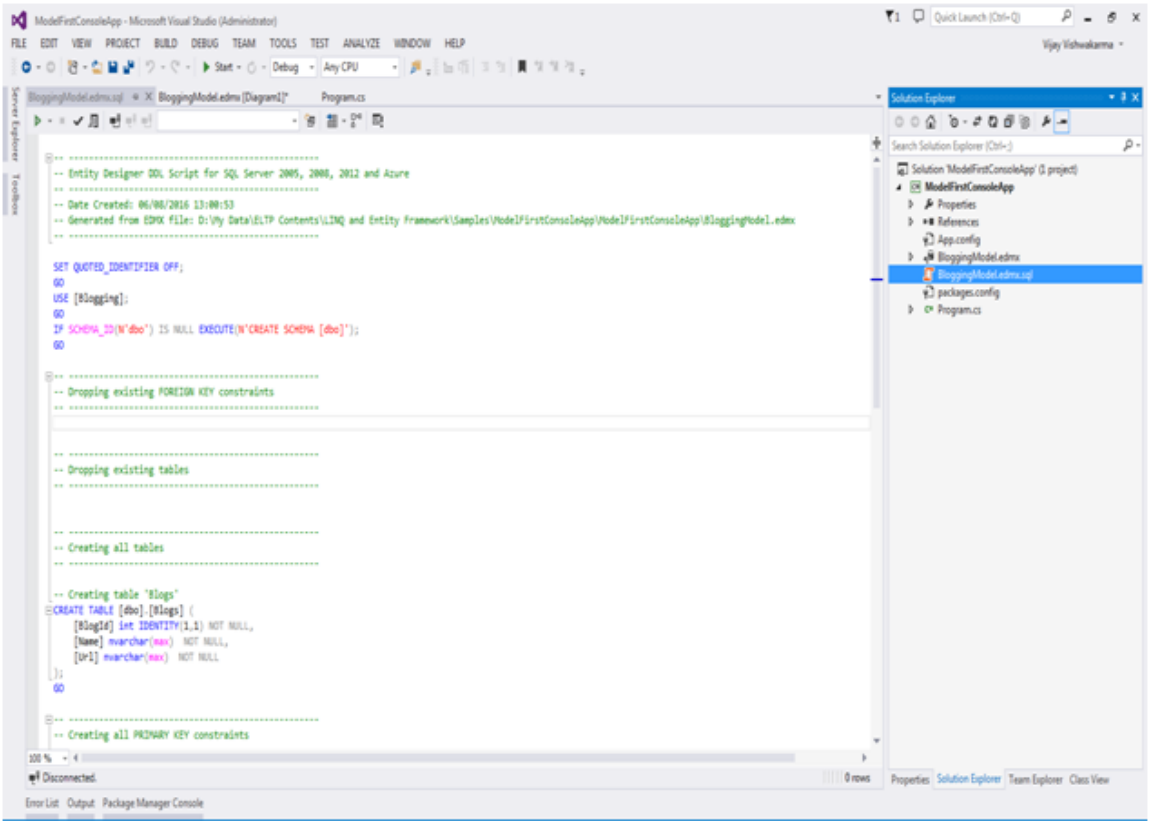
Now we have a new connection in the Generate Database Wizard



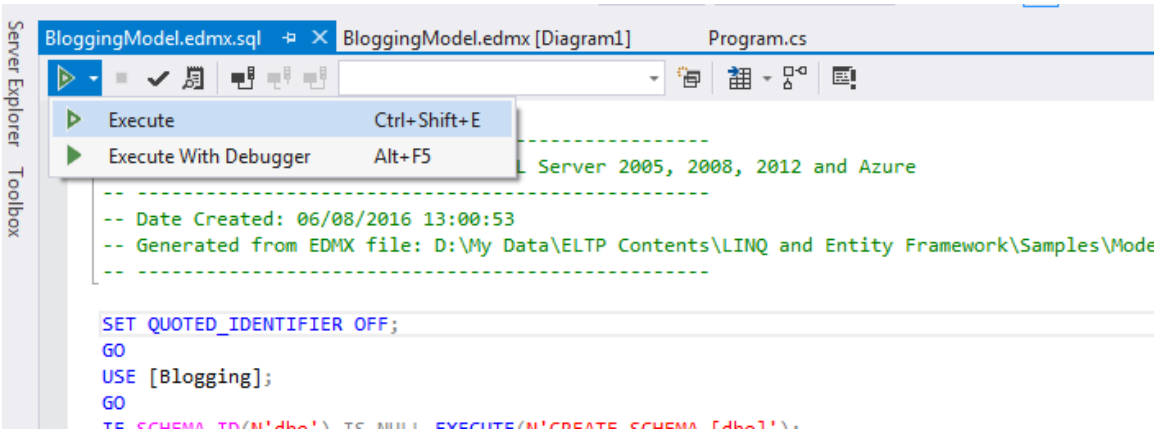
Now click on Next , DDL query for generating the Database and table will be auto generated.



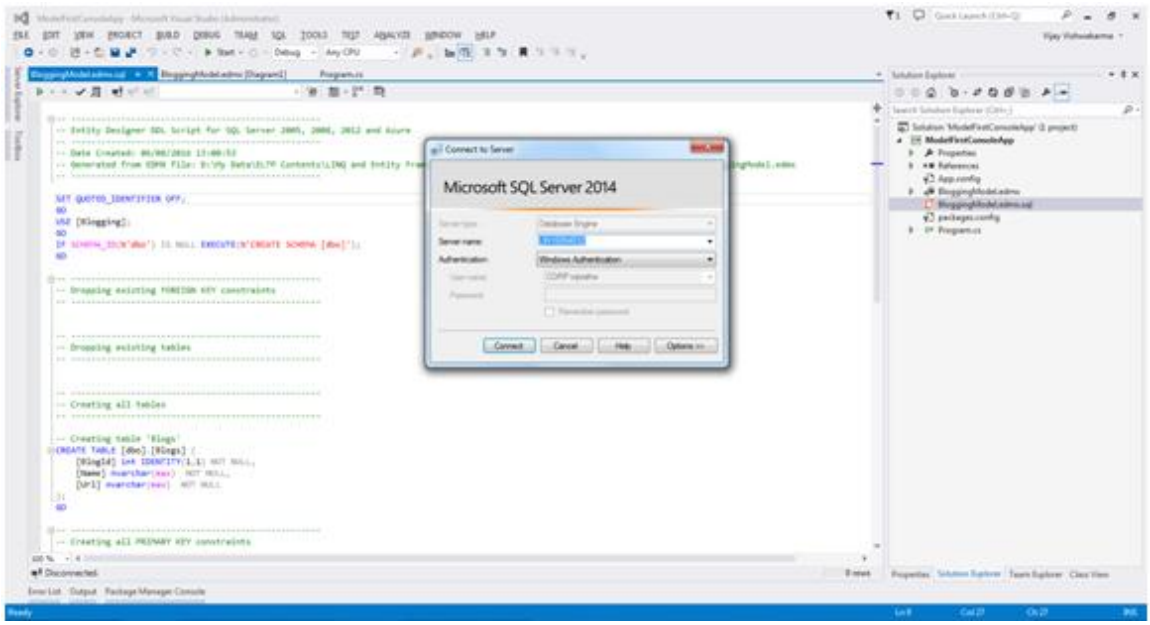
Now click on finish , a file named BloggingModel.edmx.sql will be created which will contain sql shown in the dialog for creating the database.



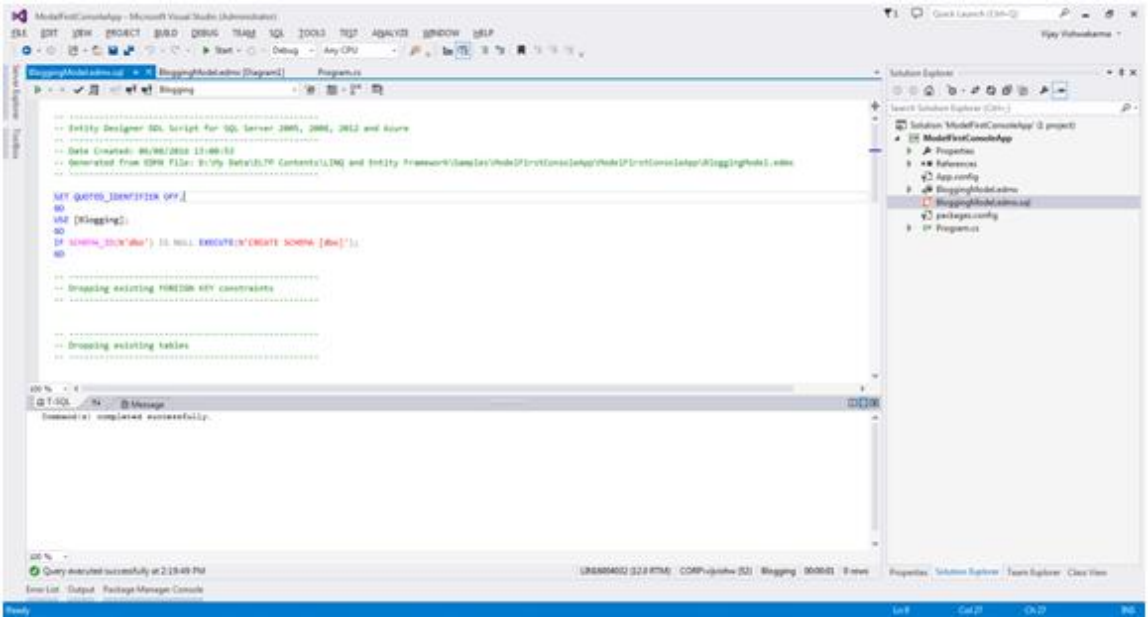
Now In the BloggingModel.edx.sql file select Execute and click on it



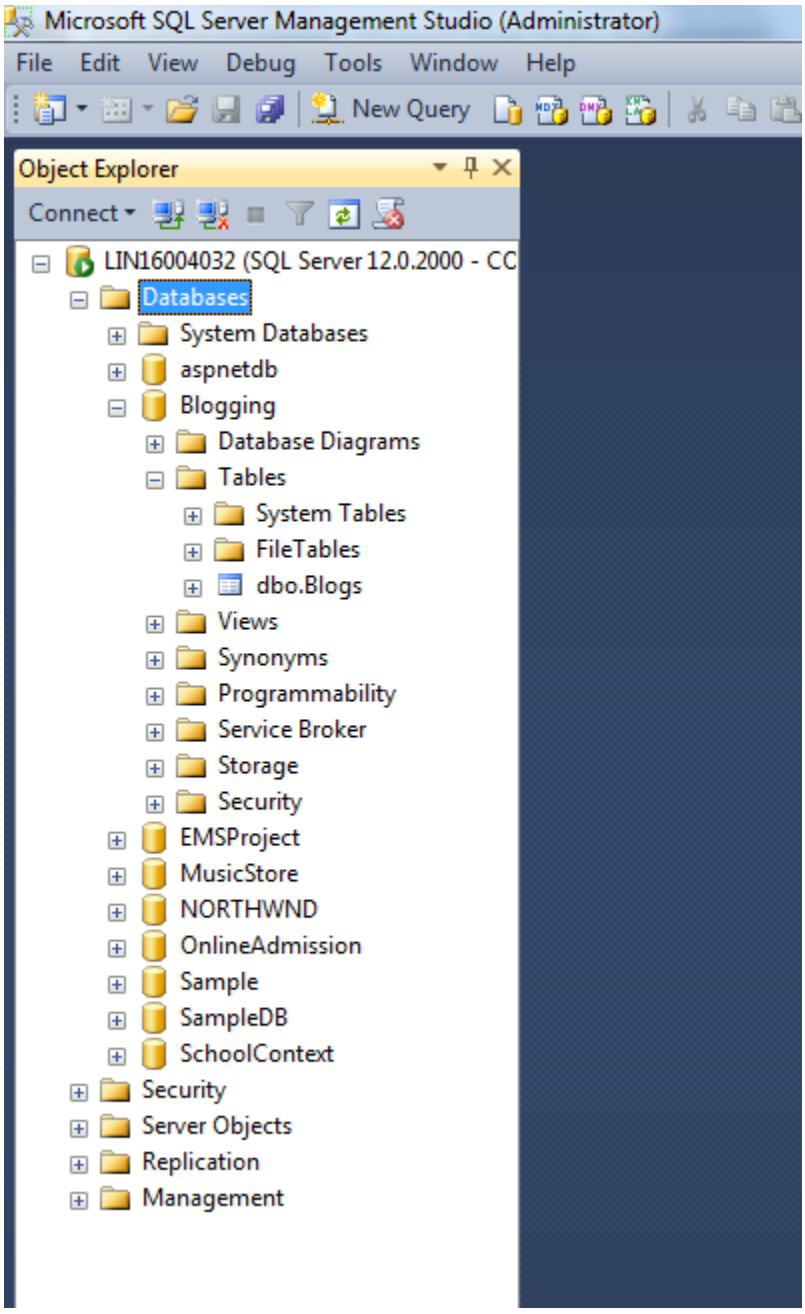
This will execute the query against the connected database instance if not connected you will be prompted to connect to sql server instance.



Click on connect and query will executed



Now in the Sql Server Management Studio we can the table which has been created



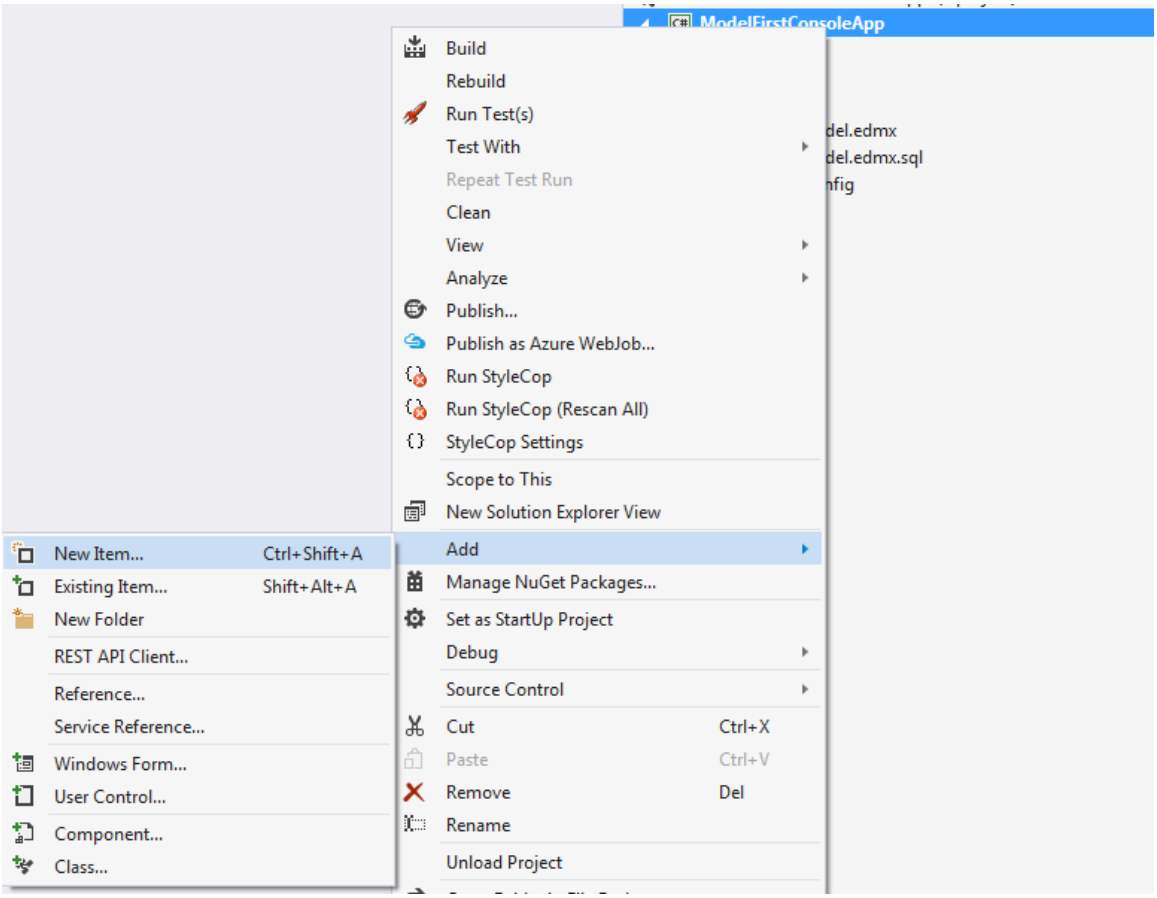
Complex Type:-

Complex types are non-scalar properties of entity types that enable scalar properties to be organized within entities. Like entities, complex types consist of scalar properties or other complex type properties.

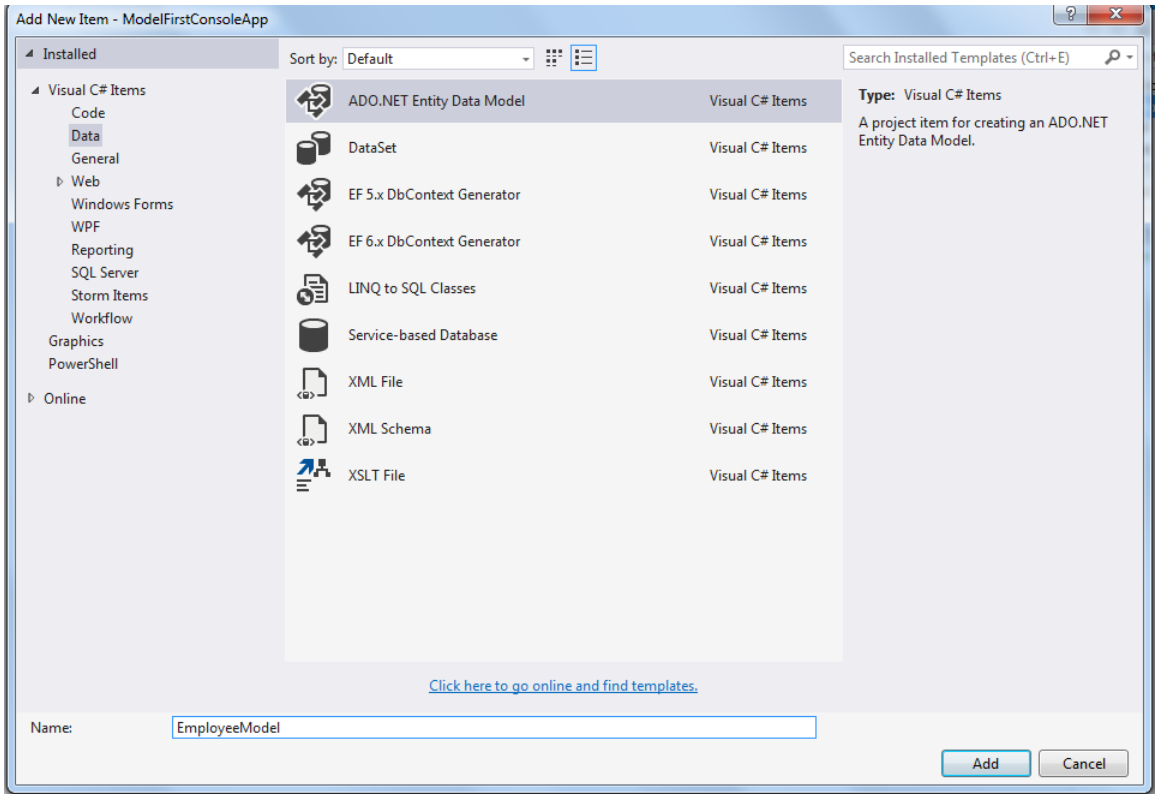
Solution:-

Creating a Complex Type using EF Designer.

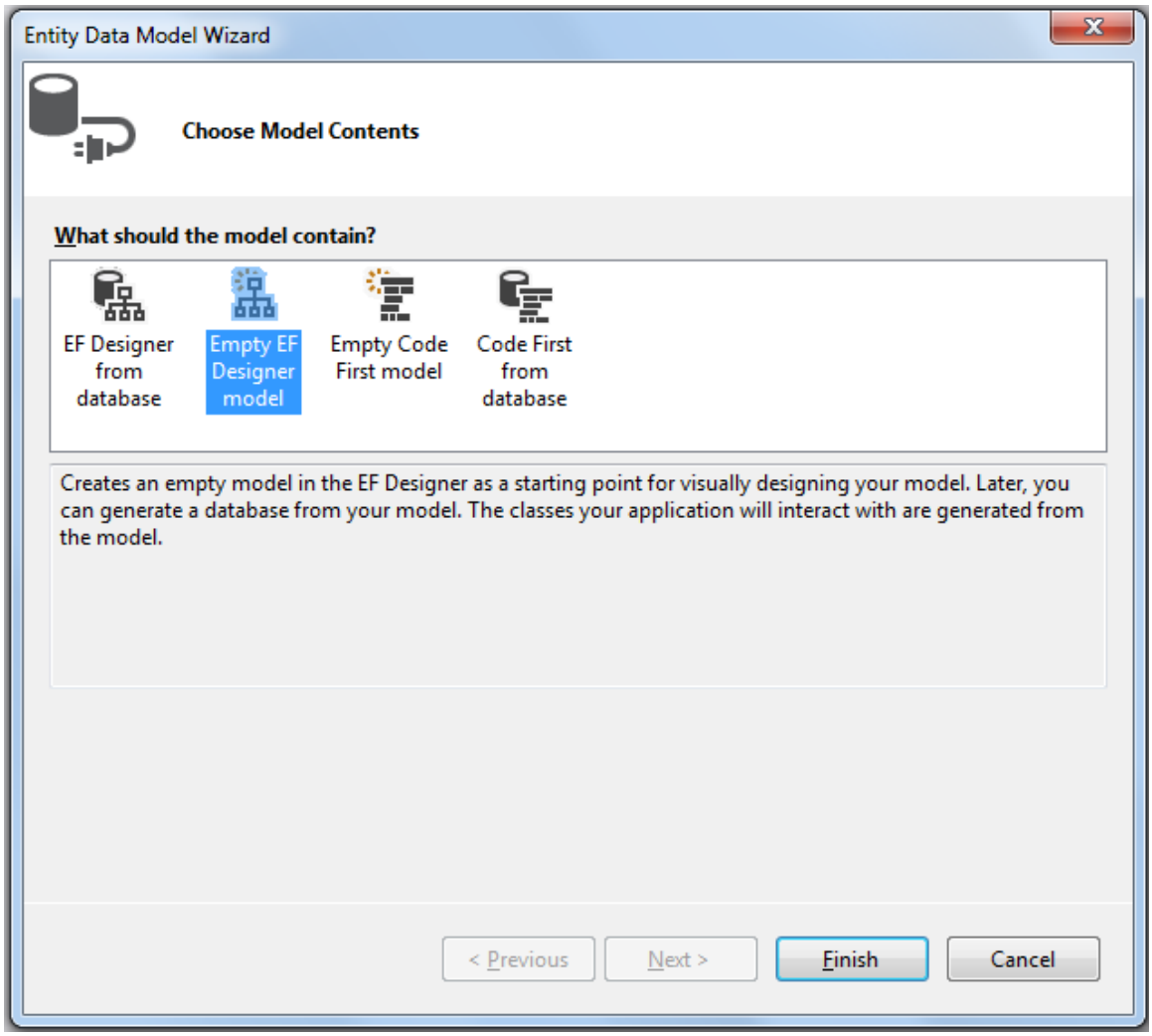
In the ModelFirstConsoleApp Example, Right click on the project in solution explorer then click on Add→New Item



In the Add New Item Dialog select ADO.Net Entity Data Model and Name it as EmployeeModel and click on Add



In the Entity Data Model Wizard Select Empty EF Designer Model and click on Finish.



Now after adding the EmployeeModel.edmx file , In the Model Browser windows right click on the Complex Types folder of EmployeeModel and Click on Add new Complex Type and Name it as AddressType

Now after creating the type Now we have to add Scalar property to it .

For Adding scalar property to the complex type right click on the complex type then Add→Scalar Property →Select data type of the property and then named the property as StreetName.

Now Similarly Three More property to the Complex Type as follows

1) Locality

2) City

3) Pincode

Now we will add a Entity to the Designer. Right click on the designer click on Add New → Entity and name the entity as Employee

Now in the Entity add a Scalar property named as Name and DOB

Now to Add the Complex type to the Entity right click on the entity and click on Add New →Complex Property and name the property as Address

As we have only one Complex Type in the Model so the newly added comple property in the Entity is Model to that complex type

Now we have to generate database from the model for that Right click on the designer and click on the Generate Database from Model.

In this we will use the connection that we have created for Blogging database.

Click on Finish this will create a Sql file named as EmployeeModel.edmx.sql which we will execute against database for creating database and tables .

Now click on the Execute button to execute the sql query against the database.

Connect to database

Now to view the table open Sql Server Management Studio and in the Object Explorer expand blogging database and then expand table tab to view the newly created table.

Lab 4. Basic Query Operations using LINQ to Entities

Goals	**Understand the process of performing LINQ queries on Entity Models Learn to use of LINQ to Entities Learn to Manipulate Data**
Time	**60 Minutes**

Solution:-

Open Visual studio and create a new console application named as LINQEFConsoleApp and the Entity Framework Library using the Nuget Package Manager to the application.

Now we have to create a Entity Data Model from an existing database.

Right click on the project in the solution explorer and select Add→ New Item

In the new Item Dialog Box select ADO.Net Entity Data Model and name it as Training Model

In the Entity Data Model Wizard select EF Designer from database and click on Next and configure the datasource.

Now Click on New Connection in the Choose Your Data source option

In the connection properties windows Provide the sql server name , authentication type and database name click on Test Connection to test the connection .

Click on OK

Now click on next to select the database objects which will be part of Entity Data Model.

Tick the Staff_Master and Student_Master Table and click on Finish this will add the entities to the model.

Now we have to write LINQ query against the model for reading the data.

Write Linq queries for the following.

1) To display staff details write the following query
2) To display list of employee whose salary is more than 30000
   Perform the following query by yourself
3) Display the list of student where city is not null
4) Display the list of student which includes Student name, department and date of birth
5) Display count of total student belonging to Bangalore
6) Display list of employees whose salary is more than the average salary of the employee.

Data Manipulation:-

Now we will CRUD operation on the model that we have create . we will use Student_Master Entity.

To ADD a record to the student master entity write the following code

Output :-

To Update a Record Add the following code

Output :-

To Delete a Record Add the following code

Output:-

To-Do Assignments

1) Create a Console application to perform CRUD operation . You have to perform following step.
    a. Create a table named Employee which will have the following fields
        i. ID
        ii. Name
        iii. DOB
        iv. DOJ
        v. Designation
        vi. Salary
    b. Add a Entity Data Model to project which will include above mentioned Entity.
    c. Using LINQ to Entities write the following functionality and execute them
        i. Add a Employee details
        ii. Updating a Employee details
        iii. Searching for an Employee based on It ID
        iv. Deleteing an employee.

