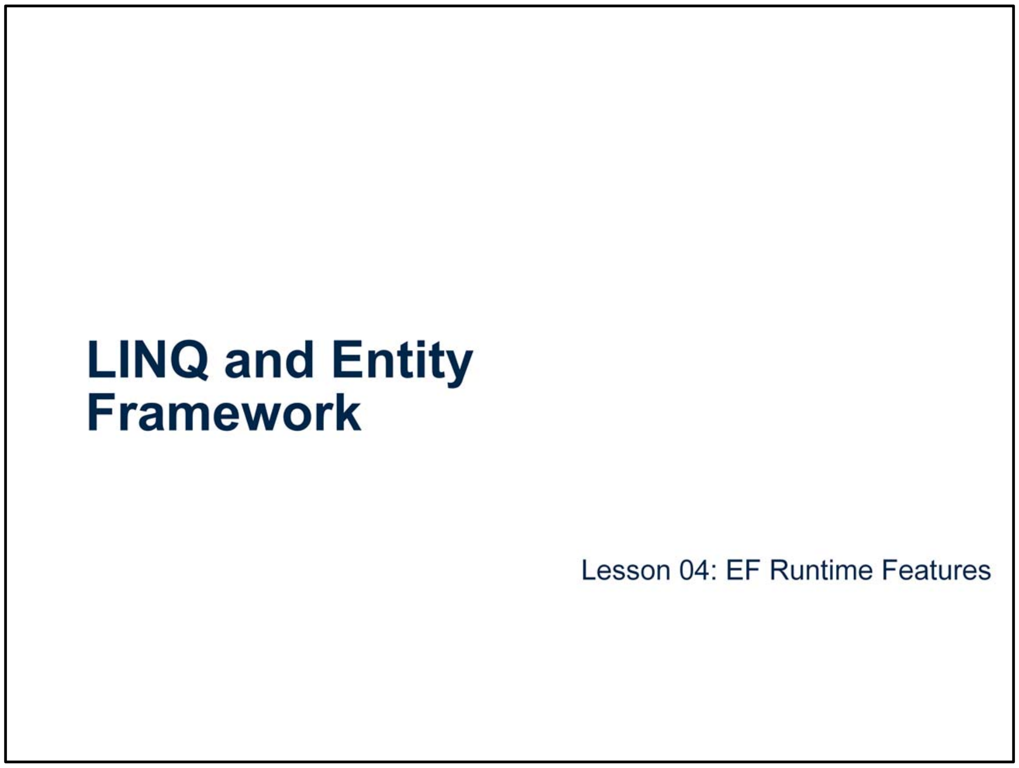
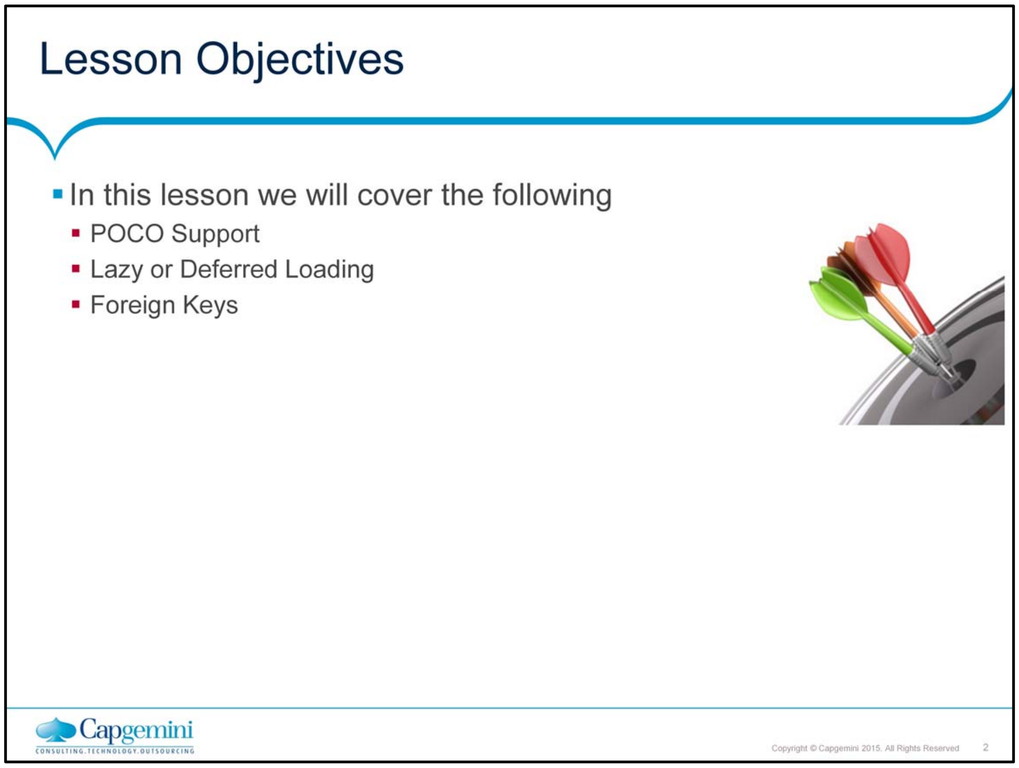


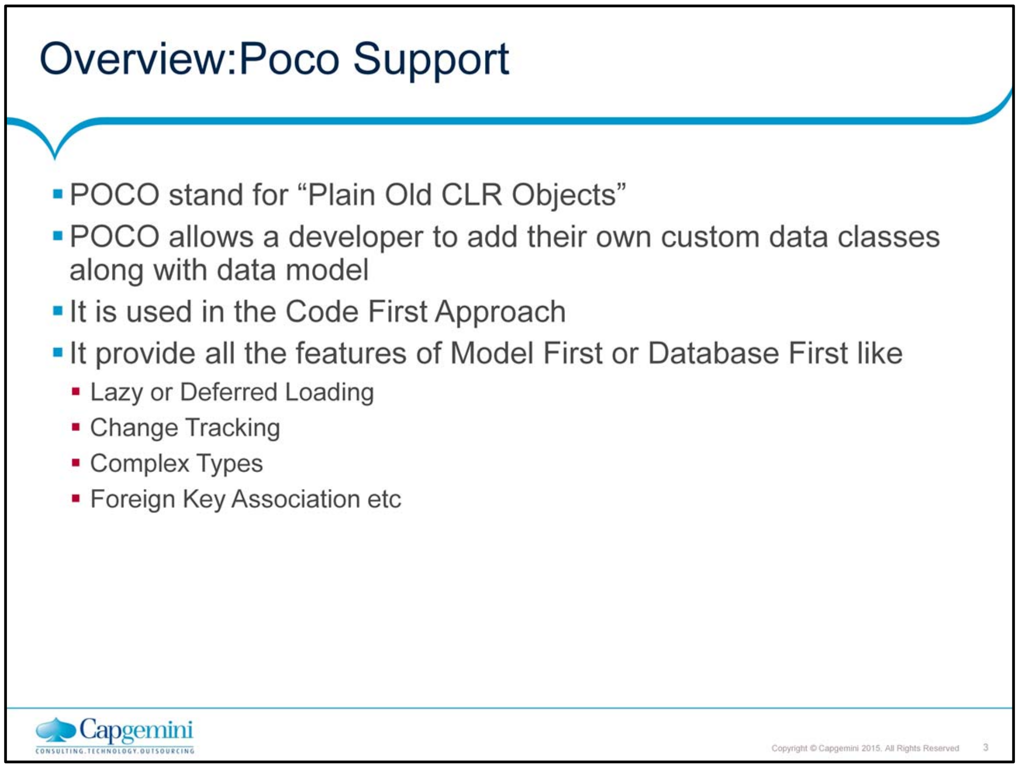
LINQ and Entity Framework

Lesson 04: EF Runtime Features



Lesson Objectives

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



Overview:Poco Support

- POCO stand for “Plain Old CLR Objects”
- POCO allows a developer to add their own custom data classes along with data model
- It is used in the Code First Approach
- It provide all the features of Model First or Database First like
  - Lazy or Deferred Loading
  - Change Tracking
  - Complex Types
  - Foreign Key Association etc

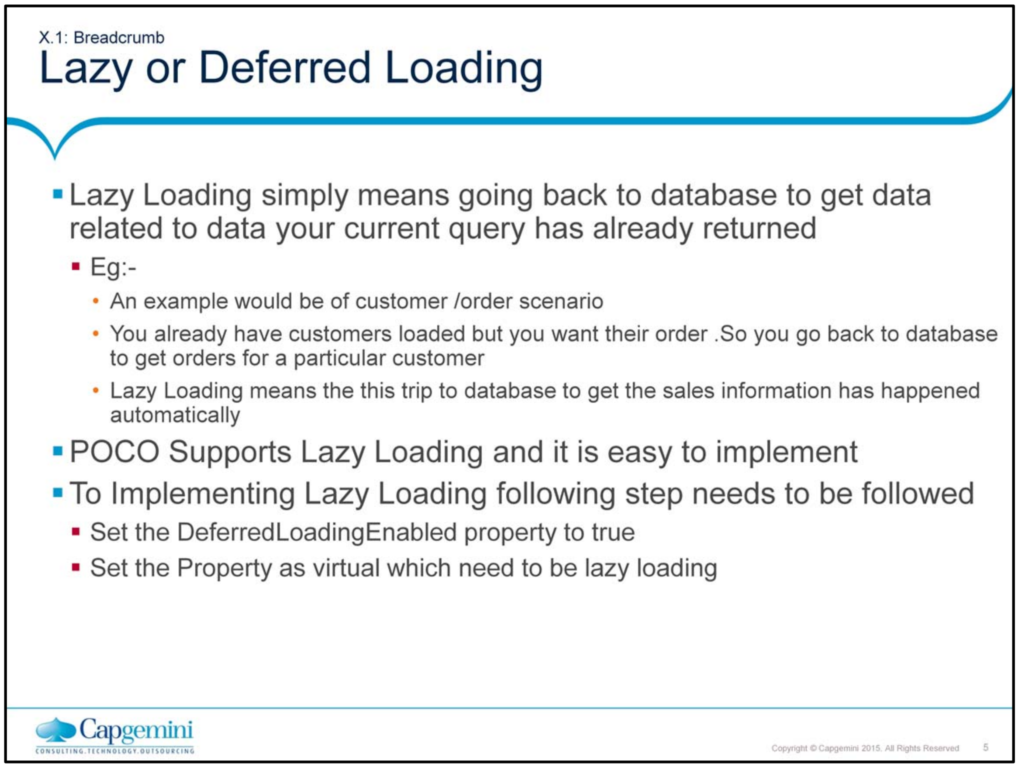
Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

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; }
}
```


X.1: Breadcrumb

Lazy or Deferred Loading

- Lazy Loading simply means going back to database to get data related to data your current query has already returned
  - Eg:-
    - An example would be of customer /order scenario
    - You already have customers loaded but you want their order .So you go back to database to get orders for a particular customer
    - Lazy Loading means the this trip to database to get the sales information has happened automatically
- POCO Supports Lazy Loading and it is easy to implement
- To Implementing Lazy Loading following step needs to be followed
  - Set the DeferredLoadingEnabled property to true
  - Set the Property as virtual which need to be lazy loading

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.

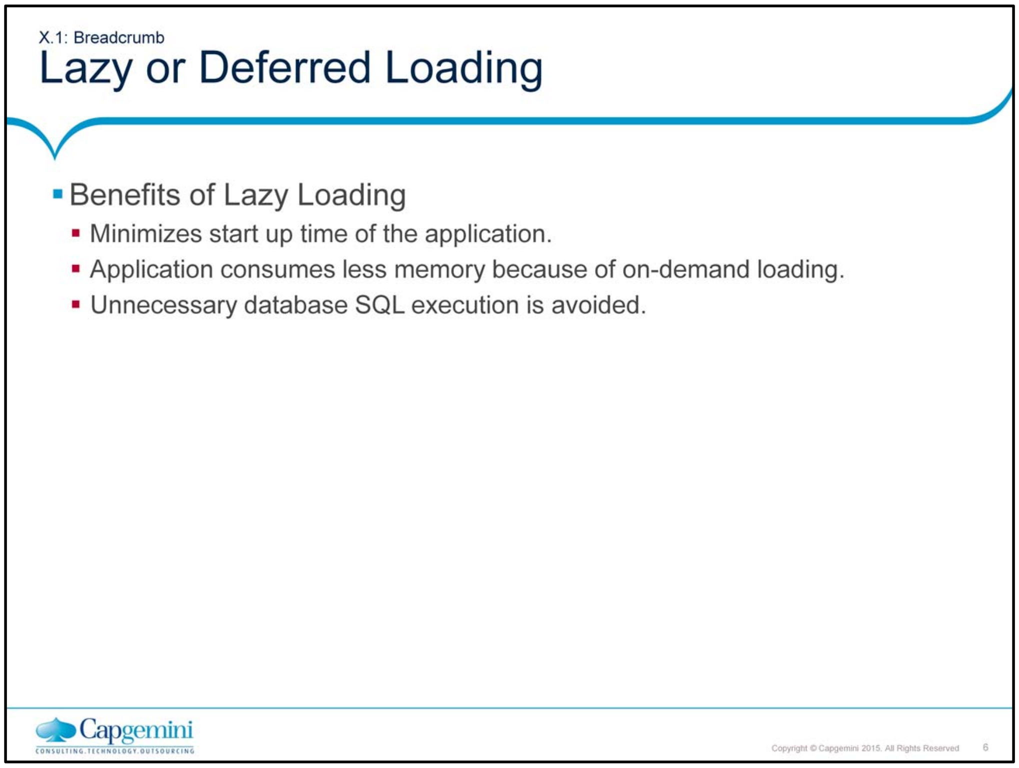
An example of this would be a customer/orders scenario. You already have the customers loaded, but you want their orders. So now you go back to the database to get orders for a particular customer. Lazy loading means that this trip to the database to get the sales information has happened
automatically.

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:-

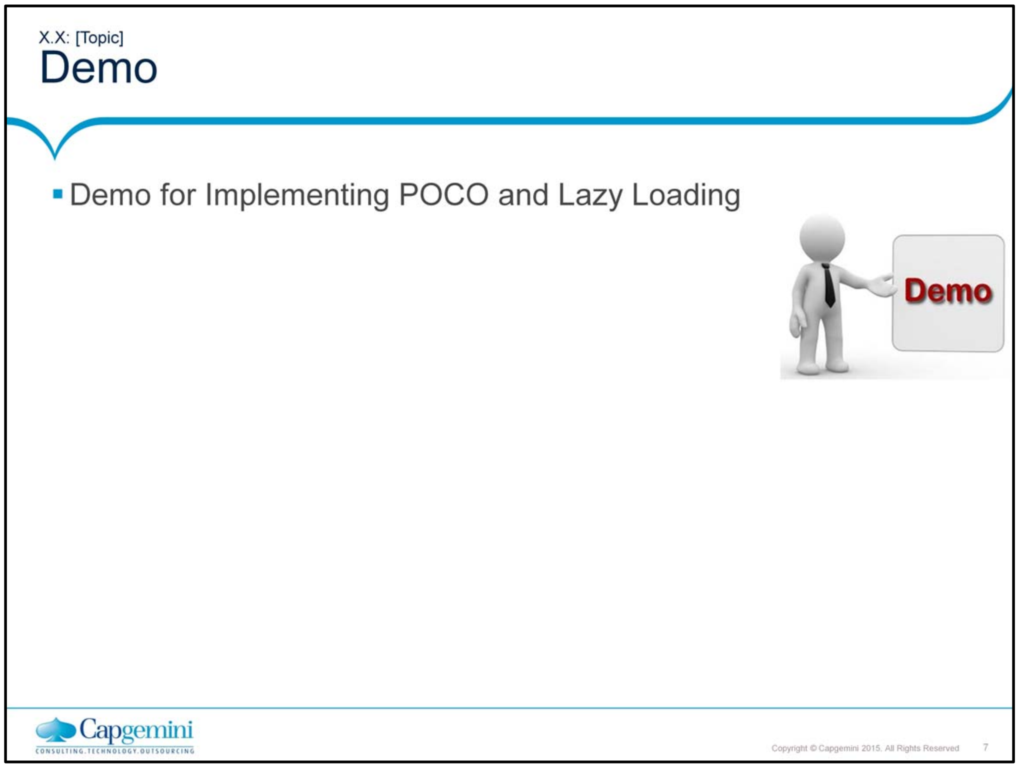
public Guid rowguid { get; set; }
public DateTime ModifiedDate { get; set; }
public virtual Contact Contact { get; set; }
public int ContactID { get; set; }



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.

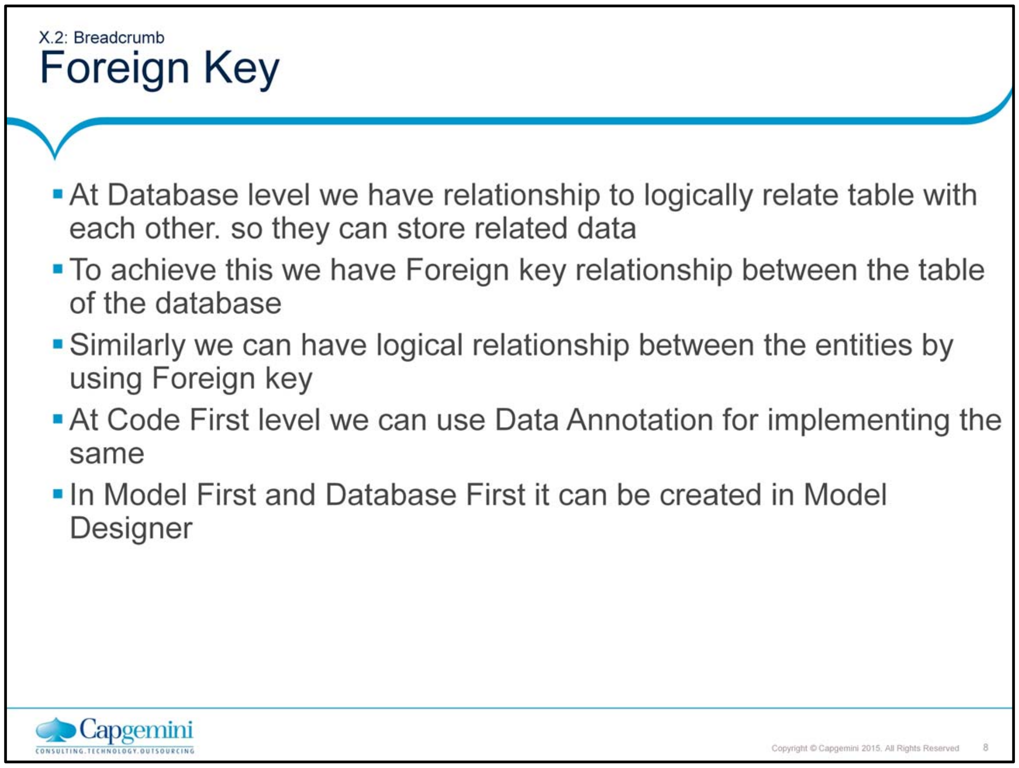


X.X: [Topic]

Demo

- Demo for Implementing POCO and Lazy Loading

Add the notes here.



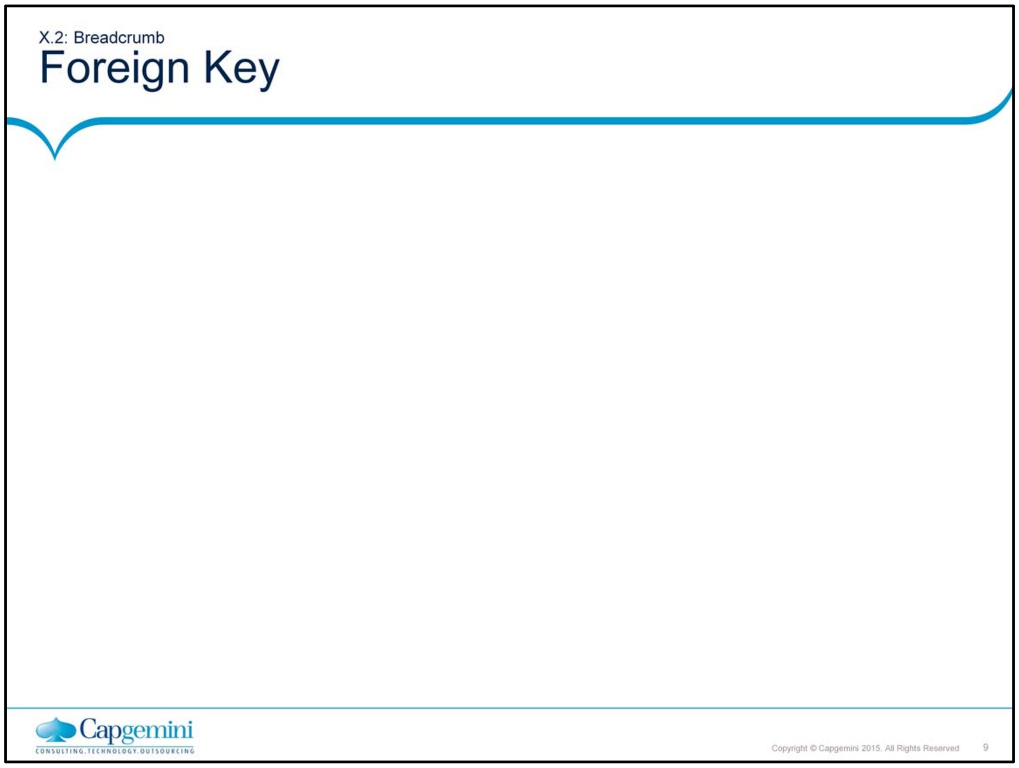
X.2: Breadcrumb

Foreign Key

- At Database level we have relationship to logically relate table with each other. so they can store related data
- To achieve this we have Foreign key relationship between the table of the database
- Similarly we can have logical relationship between the entities by using Foreign key
- At Code First level we can use Data Annotation for implementing the same
- In Model First and Database First it can be created in Model Designer

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.



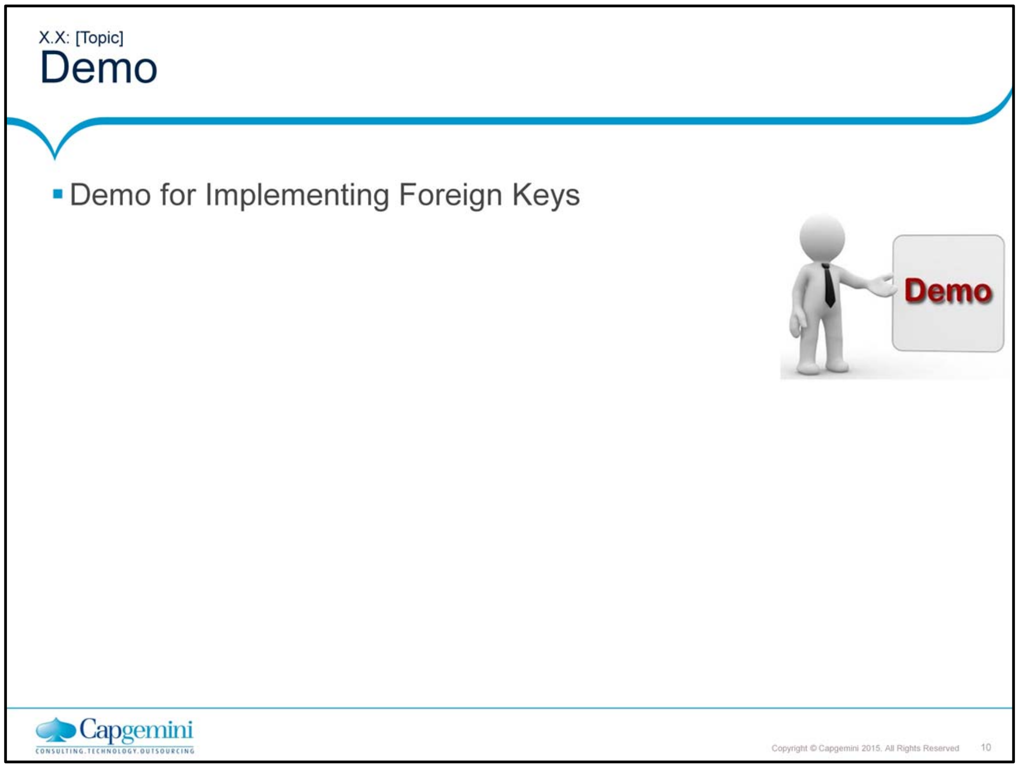
X.2: Breadcrumb

Foreign Key

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; }
}
```

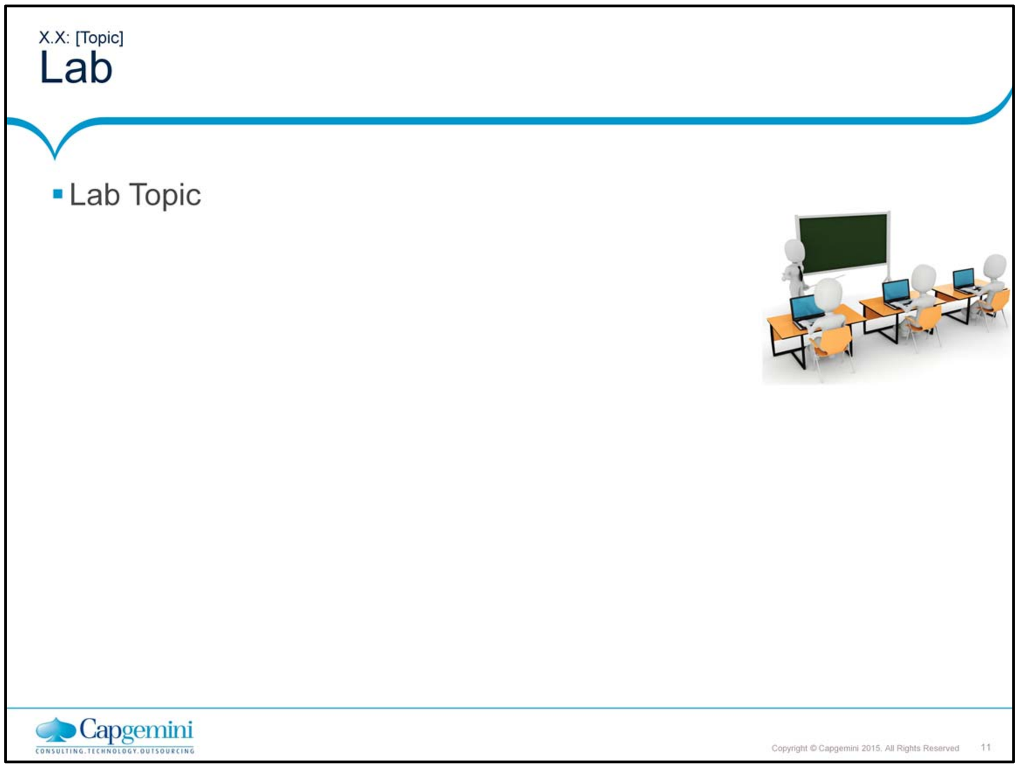


X.X: [Topic]

Demo

- Demo for Implementing Foreign Keys

Add the notes here.

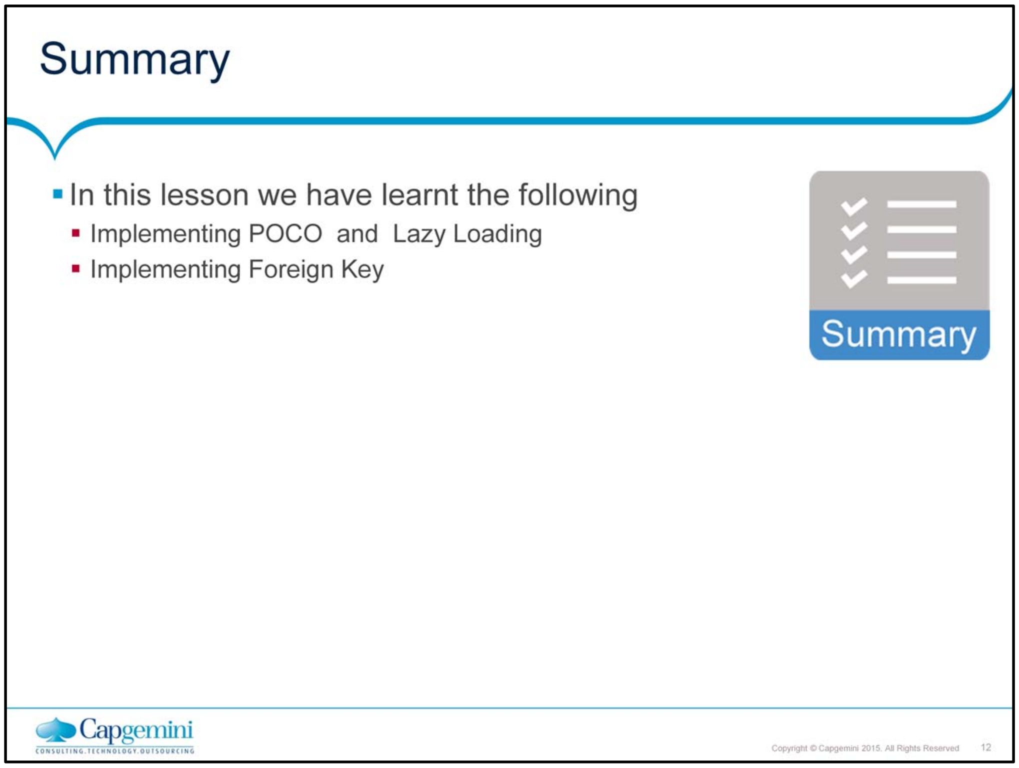


X.X: [Topic]

Lab

- Lab Topic

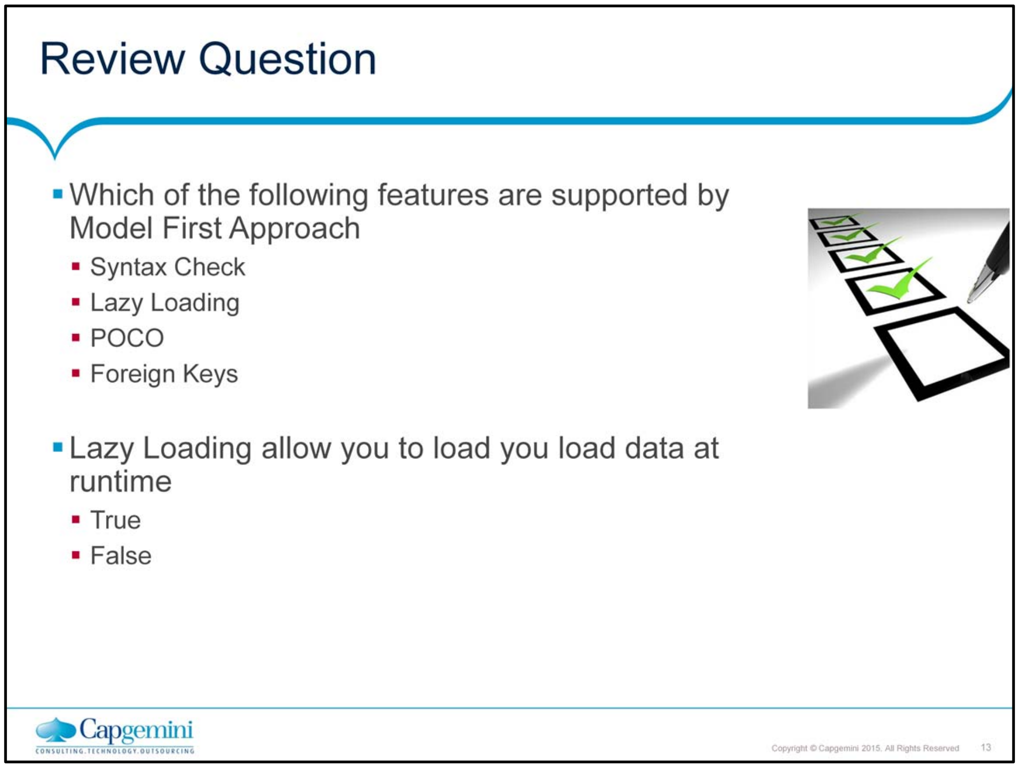
Add the notes here.



Summary

- In this lesson we have learnt the following
  - Implementing POCO and Lazy Loading
  - Implementing Foreign Key

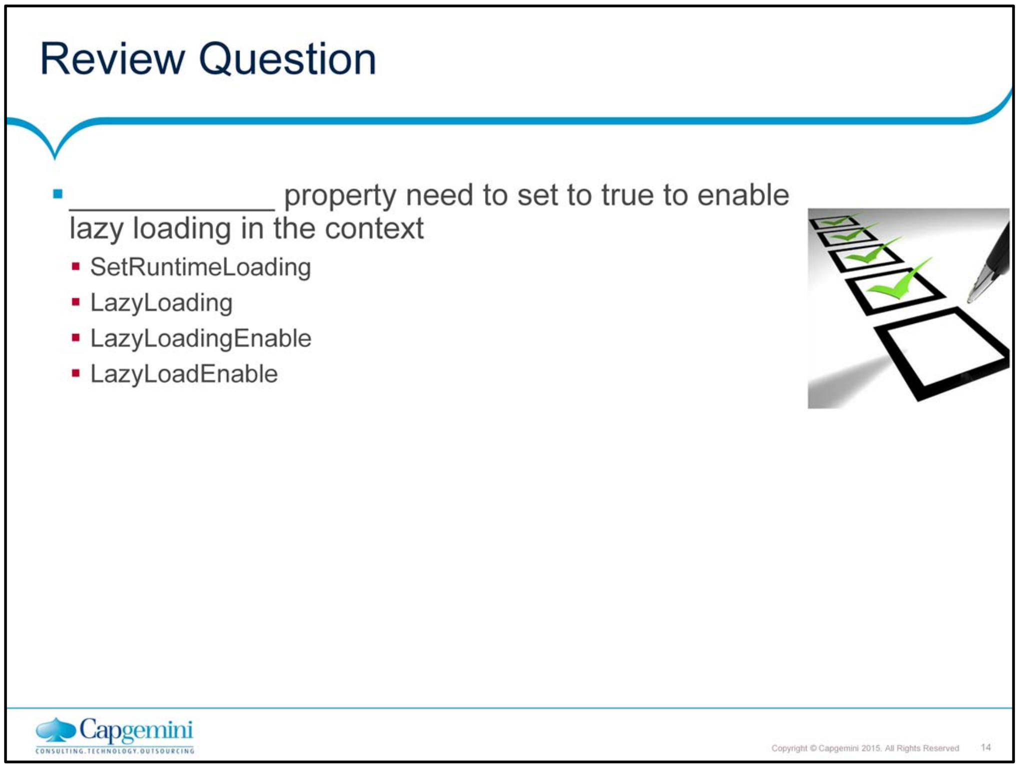
Add the notes here.



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

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.