Module 7

Accessing a Database

Module Overview

- Accessing a Database
- Creating and Using Entity Data Models
- Querying Data by Using LINQ

Lesson 1: Accessing a Database

- What's a RDBMS
- Database connection
- Connection string
- SQL command
- DataReader
  - Reads data row by row
- Entity Classes
- DataSet
  - Holds data in memory
- DataAdapter
  - Executes an SQL command

Lesson 2: Creating and Using Entity Data Models

- Introduction to the ADO.NET Entity Framework
- Using the ADO.NET Entity Data Model Tools
- Demonstration: Creating an Entity Data Model
- Customizing Generated Classes
- Reading and Modifying Data by Using the Entity Framework
- Demonstration: Reading and Modifying Data in an EDM

Introduction to the ADO.NET Entity Framework

- Why Entity Framework?
- The ADO.NET Entity Framework supports:
  - Automating changes to the data model
  - Writing code against a conceptual model
  - Easy updating of applications to a different data source
  - Writing data access code that supports compile-time type-checking
- EDMs

Using the ADO.NET Entity Data Model Tools

- Tools support:
  - Database-first design by using the Entity Data Model Wizard
  - Code-first design by using the Generate Database Wizard
- They also provide:
  - Designer pane for viewing, updating, and deleting entities and their relationships
  - Update Model Wizard for updating a model with changes that are made to the data source
  - Mapping Details pane for viewing, updating, and deleting mappings

Demonstration: Creating an Entity Data Model

In this demonstration, you will use the Entity Data Wizard to generate an EDM for an existing database

Customizing Generated Classes

- Do not modify the automatically generated classes in a model
- Use partial classes and partial methods to add business functionality to the generated classes

```
public partial class Employee
{
    partial void OnDateOfBirthChanging(DateTime? value)
    {
        if (GetAge() < 16)
        {
            throw new Exception("Employees must be 16 or over");
        }
    }
}
```

Reading and Modifying Data by Using the Entity Framework

- Reading data

```
FourthCoffeeEntities DBContext = new FourthCoffeeEntities();

// Print a list of employees.
foreach (FourthCoffee.Employees.Employee emp in
DBContext.Employees)
{
    Console.WriteLine("{0} {1}", emp.FirstName, emp.LastName);
}
```

- Modifying data

```
var emp = DBContext.Employees.First(e => e.LastName ==
"Prescott");
if (emp != null)
{
    emp.LastName = "Forsyth";
    DBContext.SaveChanges();
}
```

Demonstration: Reading and Modifying Data in an EDM

In this demonstration, you will use the **ObjectSet(TEntity)** class to read and modify data in an EDM

The Entity Framework Code First Approach

- Database is autogenerated based on your POCOs

- Changes can be synchronized manually
  - Use Package Manager Console

```
Add-Migration UniqueName

Update-Database
```

- Or automatically

```
AutomaticMigrationsEnabled = false;
```

Lesson 3: Querying Data by Using LINQ

- Querying Data
- Demonstration: Querying Data
- Querying Data by Using Anonymous Types
- Demonstration: Querying Data by Using Anonymous Types
- Forcing Query Execution
- Demonstration: Retrieving and Modifying Grade Data Lab

Querying Data

- Use LINQ to:
  - Select data
  - Sort and filter
  - Aggregate
  - Join tables
- Use LINQ to query a range of data sources, including:
  - .NET Framework collections
  - SQL Server databases
  - ADO.NET data sets
  - XML documents

Demonstration: Querying Data

In this demonstration, you will use LINQ to Entities to query data

Querying Data by Using Anonymous Types

Anonymous types

```
var s = new { Name = "Stefan", IsTeacher = true};
```

Use LINQ and anonymous types to:

- Filter data by column
- Group data
- Aggregate data
- Navigate data

Demonstration: Querying Data by Using Anonymous Types

In this demonstration, you will use LINQ to Entities to query data by using anonymous types

Forcing Query Execution

- Deferred query execution—default behavior for most queries
- Forced query execution—overrides deferred query execution:
  - **ToArray**
  - **ToDictionary**
  - **ToList**

```
IList<Employee> emp = (from e in FCEntities.Employees
                       orderby e.LastName
                       select e).ToList();
```

- IQueryable
- Immediate query execution—default behavior for queries that return a singleton value

Demonstration: Retrieving and Modifying Grade Data Lab

In this demonstration, you will learn about the tasks that you will perform in the lab for this module.

Lab: Retrieving and Modifying Grade Data

- Exercise 1: Creating an Entity Data Model from The School of Fine Arts Database
- Exercise 2: Updating Student and Grade Data by Using the Entity Framework
- Exercise 3: Extending the Entity Data Model to Validate Data

Estimated Time: 75 minutes

Lab Scenario

You have been asked to upgrade the prototype application to use an existing SQL Server database. You begin by working with a database that is stored on your local machine and decide to use the Entity Data Model Wizard to generate an EDM to access the data. You will need to update the data access code for the Grades section of the application, to display grades that are assigned to a student and to enable users to assign new grades. You also decide to incorporate validation logic into the EDM to ensure that students cannot be assigned to a full class and that the data that users enter when they assign new grades conforms to the required values.

Module Review and Takeaways

- Review Questions