Tipo : Guía de laboratorio
Capítulo : Repositorio de datos
Duración : 60 minutos

I. OBJETIVO

Implementar el patrón unit of work con el proyecto DataAccess.

II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio:

- Windows 8.1 o superior
- SQL Server 2016
- Visual Studio 2017

III. EJECUCIÓN DEL LABORATORIO

- **Tarea 1**

  Implementar el patrón unit of work con el proyecto DataAccess.

  Actividades

  1. Abrir la solución del ejercicio anterior.

  2. En la raíz del proyecto DataAccess, crear la interfaz llamada: IUnitOfWork.cs y agregar el siguiente código:

```
using DataAccess.Repositories;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataAccess
{
    public interface IUnitOfWork : IDisposable
    {
        IArtistRepository Artists { get; }
        int Complete();
    }
}
```

  3. Para implementar la interfaz creamos en la raíz del proyecto DataAccess una clase llamada: UnitOfWork.cs, en la cual deberemos insertar el siguiente código:

```
using DataAccess.Repositories;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataAccess
```

```
{
    public class UnitOfWork : IUnitOfWork
    {
        private readonly ChinookContext _context;

        public UnitOfWork(ChinookContext context)
        {
            _context = context;
            Artists = new ArtistRepository(_context);
        }

        public IArtistRepository Artists { get; private set; }

        public int Complete()
        {
            return _context.SaveChanges();
        }

        public void Dispose()
        {
            _context.Dispose();
        }
    }
}
```

4. Como procedimiento final realizaremos la actualización a nuestro proyecto de test para que use nuestro Unit of Work en lugar del repositorio inicial que creamos. Ahora nuestra clase de test debe de quedar de la siguiente manera:

```
using DataAccess.Repositories;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Models;
using System.Linq;

namespace DataAccess.Test
{
    [TestClass]
    public class ArtistRepositoryTest
    {
        private readonly UnitOfWork _unitOfWork;
        public ArtistRepositoryTest()
        {
            _unitOfWork = new UnitOfWork(new ChinookContext());
        }

        [TestMethod]
        public void Test_Connection_And_Count_Greater_Than_Zero()
        {
            var count = _unitOfWork.Artists.Count();
            Assert.AreEqual(count > 0, true);
        }

        [TestMethod]
        public void Search_Artist_By_Id()
        {
            var artist = _unitOfWork.Artists.GetById(1);
            var expectedArtist = new Artist
            {
                ArtistId = 1,
                Name = "AC/DC"
            };
            Assert.AreEqual(expectedArtist.ArtistId, artist.ArtistId);
```

```
            Assert.AreEqual(expectedArtist.Name, artist.Name);
        }

        [TestMethod]
        public void Get_List_Of_Artist()
        {
            var artistList = _unitOfWork.Artists.GetAll();
            Assert.AreEqual(artistList.Count() > 0, true);
        }

        [TestMethod]
        public void Get_List_Of_Artist_by_Store()
        {
            var artistList =
_unitOfWork.Artists.GetListArtistByStore();
            Assert.AreEqual(artistList.Count() > 0, true);
        }

        [TestMethod]
        public void Insert_Artist()
        {
            var newArtist = new Artist
            {
                Name = "Test Unit Of Work"
            };
            _unitOfWork.Artists.Add(newArtist);
            _unitOfWork.Complete();
            var expectedArtist = _unitOfWork.Artists.GetByName("Test
Unit Of Work");
            Assert.AreEqual(expectedArtist.ArtistId > 0, true);
            Assert.AreEqual(expectedArtist.Name, "Test Unit Of Work");
        }

        [TestMethod]
        public void Delete_Artist_By_Id()
        {
            var removeArtist = _unitOfWork.Artists.GetByName("Test
Unit Of Work");
            _unitOfWork.Artists.Remove(removeArtist);
            Assert.AreEqual(_unitOfWork.Complete()> 0, true);
        }
    }
}
```

5. Ejecutamos los test y validamos que los status sean **PASSED**.

IV. EVALUACIÓN

1. ¿Por qué implementar el patrón repositorio?