BÀI THỰC HÀNH SỐ 2 VỀ ENTITY FRAMEWORK

Nội dung:

- Xây dựng và phát triển dự án mã nguồn dựa trên EF, sau đó tích hợp vào website ASP.NET
- Nắm vững các tương tác cần thiết với database

Tạo 1 dự án trống tên là **CodeProject**

Câu 1. Thêm dự án Library (.DLL) tên là **EntityModel,** sau đó thực hiện gieo mã nguồn từ database cho dự án này thông qua mô hình code.

a. Tạo 1 database với 2 bảng Blog và Post như ở Lab 1.

b. Thêm Entity Framework và thư viện Dynamic LINQ vào dự án. (xem slide bài giảng)

c. Gieo mã nguồn từ database với Context tên là EF.

d. Build project này để cho ra thư viện tên là **EntityModel.dll**

Câu 2. Thêm 1 project mới vào dự án Model tên là **EntityController**.

a. Thêm thư viện **EntityModel.dll** vào dự án.

b. Tạo thư mục Core và Extra, trong thư mục Core tạo file code tên là **Base.cs** với nội dung

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using EntityModel;

namespace EntityController
{
    public abstract class Base
    {
        public TDAEntities db;
        public Base()
        {
            db = new TDAEntities();
        }
    }
}
```

c. Tạo 2 file tên là **PostController.cs** trong thư mục **Core** với nội dung:

```
using System;
using System.Collections.Generic;
using System.Data;
```

```
using System.Linq;
using System.Linq.Dynamic;
using System.Text;
using System.Threading.Tasks;
using Model;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using Entity = EntityModel.Post;

namespace EntityController
{
    public partial class PostController : Base
    {
        public DbSet<Entity> Execute;
        public PostController()
        {
            Execute = db.Posts;
        }

        #region Common execution
        /// <summary>
        /// Execute query string (if select always use * - select *) for this object only
        /// </summary>
        /// <param name="query">query</param>
        /// <returns>bool</r
        /// eturns>
        public List<Entity> ExecuteQuery(string query)
        {
            return Execute.SqlQuery(query).ToList();
        }
        #endregion

        #region Insert
        /// <summary>
        /// Insert an entity
        /// </summary>
        /// <param name="e">Entity</param>
        /// <returns>bool</returns>
        public bool Insert(Entity e)
        {
            Execute.Add(e);
            return (db.SaveChanges() > 0);
        }
        /// <summary>
        /// Insert entity list
        /// </summary>
        /// <param name="list">Entity list</param>
        /// <returns>bool</returns>
        public bool Insert(List<Entity> list)
        {
            foreach (Entity e in list)
            {
                Execute.Add(e);
            }
            try { db.SaveChanges(); }
            catch { return false; }
            return true;
        }
```

```
        #endregion

        #region Delete queries

        /// <summary>
        /// Delete by conditions, not use for Like operator
        /// </summary>
        /// <param name="conditions"></param>
        /// <returns></returns>
        public bool DeleteWhere(string conditions)
        {
            Execute.RemoveRange(Execute.AsQueryable().Where(conditions).ToList());
            try { db.SaveChanges(); }
            catch { return false; }
            return true;
        }

        /// <summary>
        /// Delete class object
        /// </summary>
        /// <param name="l">Class object</param>
        /// <returns>bool</returns>
        public bool Delete(Entity e)
        {
            Execute.Attach(e);
            Execute.Remove(e);
            try { db.SaveChanges(); }
            catch { return false; }
            return true;
        }

        /// <summary>
        /// Delete object list
        /// </summary>
        /// <param name="list"></param>
        /// <returns></returns>
        public bool Delete(List<Entity> list)
        {
            foreach (Entity e in list)
            {
                Execute.Attach(e);
                Execute.Remove(e);
            }
            try { db.SaveChanges(); }
            catch { return false; }
            return true;
        }
        #endregion

        #region Select queries
        /// <summary>
        /// Select by conditions, not use for Like operator
        /// </summary>
        /// <param name="conditions"></param>
        /// <returns></returns>
        public List<Entity> SelectWhere(string conditions)
        {
            return Execute.AsQueryable().Where(conditions).ToList();
```

```
        }

        /// <summary>
        /// Select by conditions and sort by orders, not use for Like operator
        /// </summary>
        /// <param name="conditions">conditions</param>
        /// <param name="orders">orders</param>
        /// <returns></returns>
        public List<Entity> SelectOrderWhere(string conditions, string orders)
        {
            return Execute.AsQueryable().Where(conditions).OrderBy(orders).ToList();
        }

        /// <summary>
        /// Select all records
        /// </summary>
        /// <returns>List</returns>
        public List<Entity> SelectAll()
        {
            return Execute.ToList();
        }

        /// <summary>
        /// Select all by orders (Order = "LinkID ASC, LinkURL DESC")
        /// Use Linq.Dynamic library
        /// </summary>
        /// <param name="Order">Orders string</param>
        /// <returns>List</returns>
        public List<Entity> SelectAll(string orders)
        {
            return Execute.AsQueryable().OrderBy(orders).ToList();
        }

        /// <summary>
        /// Select top
        /// </summary>
        /// <param name="number">the number of records</param>
        /// <returns>List</returns>
        public List<Entity> SelectTop(int number)
        {
            return Execute.Take(number).ToList();
        }

        /// <summary>
        /// Select top by orders
        /// </summary>
        /// <param name="number">the number of records</param>
        /// <param name="orders">orders' string - example: LinkID ASC, LinkName
DESC</param>
        /// <returns>List</returns>
        public List<Entity> SelectTop(int number, string orders)
        {
            return Execute.AsQueryable().OrderBy(orders).Take(number).ToList();
        }
        #endregion

        #region Update
        /// <summary>
```

```
        /// Update an entity
        /// </summary>
        /// <param name="e">Entity</param>
        /// <returns>bool</returns>
        public bool Update(Entity e)
        {
            Execute.Attach(e);
            db.Entry(e).State = EntityState.Modified;
            return (db.SaveChanges() > 0);
        }
        /// <summary>
        /// Update entity list
        /// </summary>
        /// <param name="list">Entity list</param>
        /// <returns>bool</returns>
        public bool Update(List<Entity> list)
        {
            foreach (Entity e in list)
            {
                Execute.Attach(e);
                db.Entry(e).State = EntityState.Modified;
            }
            try { db.SaveChanges(); }
            catch { return false; }
            return true;
        }

        #endregion
    }
}
```

d. Tiếp tục tạo file **BlogController.cs** trong thư mục Core với nội dung tương tự PostController.cs, thay đổi 1 số chỗ như sau:

```
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using Entity = EntityModel.Blog;

namespace EntityController
{
    public partial class BlogController : Base
    {
        public DbSet<Entity> Execute;
        public BlogController()
        {
            Execute = db.Blogs;
        }
        //nội dung các đoạn code khác tương tự PostController.cs
}
```

e. Build dự án Controller để sinh ra thư viện **EntityController.dll**

Câu 3. Thêm một dự án web ASP.NET mới tên là **Application** vào dự án trên.
a. Thêm các thư viện EntityController.dll, EntityModel.dll vào thư mục khởi tạo tên là **Bin**
b. Thêm Entity Framework, Dynamic LINQ vào dự án (tương tự câu 1)

Sinh viên tự do trình bày giao diện cho các câu sau:
c. Tạo trang web tên là **Blog.aspx**, hiển thị nội dung trong bảng Blog thông qua control Repeater. Thêm link **chi tiết** ở mỗi record khi nhấn vào thì hiển thị các Post thuộc Blog thông qua trang **PostList.aspx?BlogID=123** với BlogID = 123 là chỉ mục (ID) của Blog cần tìm.

```
<asp:Repeater runat="server" ID="dataRepeater"
	<ItemTemplate>
		<%# Eval("BlogID") %>
		….
	</ItemTemplate>
</asp:Repeater>
```

d. Tạo trang web tên là **Post.aspx**, hiển thị nội dung trong bảng Post thông qua control Repeater.

e. Tạo trang web tên là **AddBlog.aspx** với giao diện thêm 1 Blog mới.