Chapter 4

Integrating Windows Forms with SQL Server

(Document opens in Protected Mode.
Click on Enable Editing to get started.)

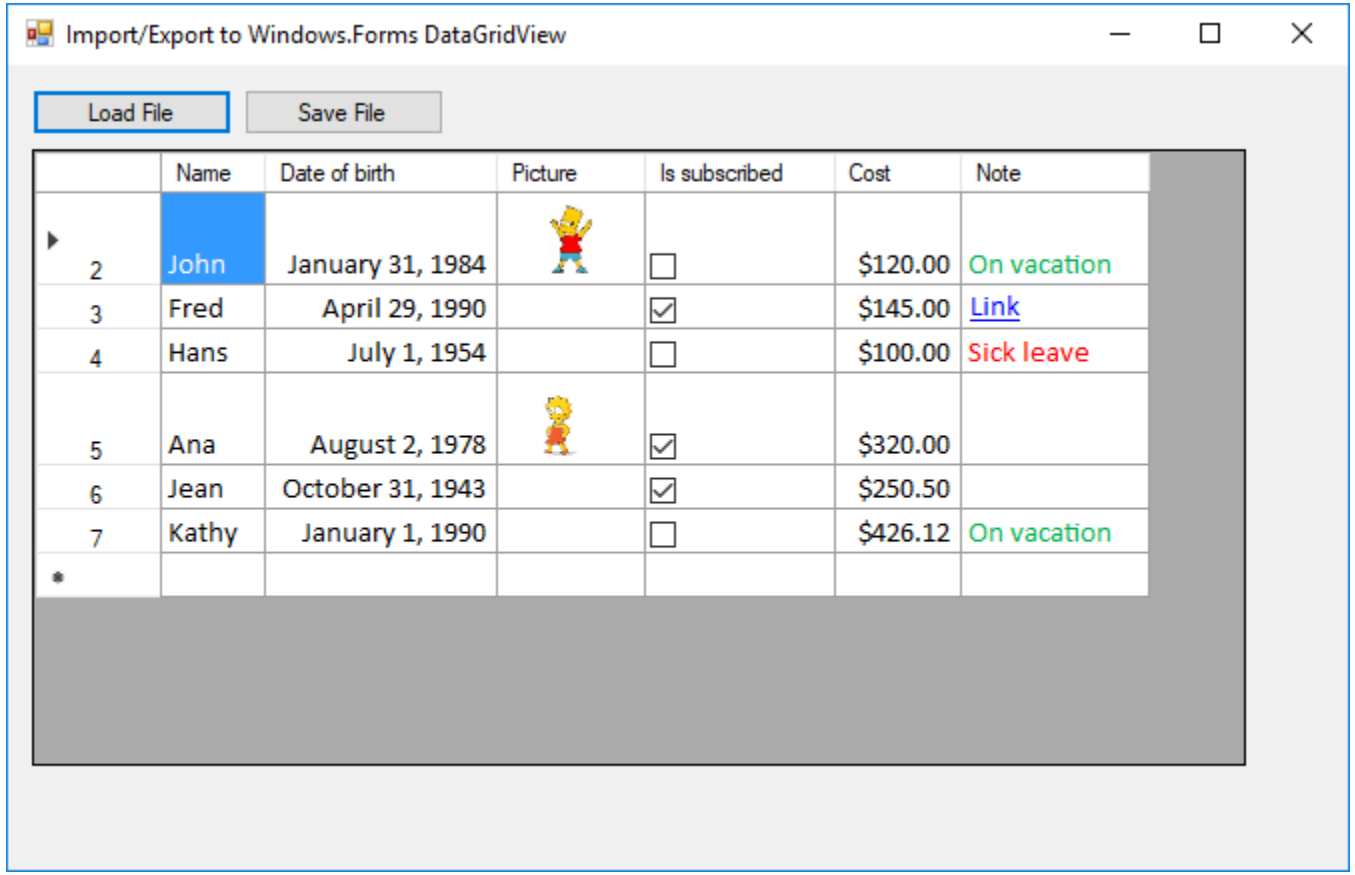
Created by Nguyen Quang Vinh

DataGridView

Control trong Windows Forms dùng để hiển thị dữ liệu dưới dạng bảng.

Các đặc điểm:

- Hiển thị danh sách dữ liệu.
- Hỗ trợ thao tác như sắp xếp, chỉnh sửa trực tiếp.
- Kết hợp tốt với ADO.NET Entity Framework.



Bài tập – Quản lý sinh viên

Xây dựng một ứng dụng Windows Forms đơn giản để quản lý thông tin sinh viên.

Chức năng:

- **Hiển thị danh sách sinh viên:** Dữ liệu được lấy từ một danh sách (List) các đối tượng SinhVien. Hiển thị lên DataGridView.
- **Thêm sinh viên mới:** Cho phép người dùng nhập thông tin sinh viên (Mã SV, Tên SV, Lớp, Điểm) và thêm vào danh sách.
- **Xóa sinh viên:** Cho phép người dùng chọn một hoặc nhiều sinh viên trên DataGridView và xóa khỏi danh sách.
- **Tìm kiếm sinh viên:** Cho phép người dùng tìm kiếm sinh viên theo Tên SV.
- **Lưu/Đọc dữ liệu:** Lưu danh sách sinh viên xuống file (ví dụ: file text, CSV) và đọc dữ liệu từ file lên khi khởi động ứng dụng.

ADO.NET Entity Framework Core

ADO.NET Entity Framework Core (EF Core) là một ORM (Object-Relational Mapping) framework trong .NET Core, giúp phép các nhà phát triển làm việc với cơ sở dữ liệu như thao tác với các đối tượng C# thay vì SQL.

Lợi ích của Entity Framework Core:

- Giảm bớt khối lượng viết code SQL tự nhiên.
- Hỗ trợ nhiều DBMS như SQL Server, MySQL, SQLite, Cosmos DB.
- Dễ bảo trì và nâng cấp.
- Hỗ trợ nhiều nền tảng (cross-platform).

Các khái niệm cơ bản trong EF Core

DbContext: Là lớp trung tâm quản lý các đối tượng và thao tác cơ sở dữ liệu.

Entity: Là các lớp đại diện cho bảng trong cơ sở dữ liệu.

LINQ to Entities: Là ngôn ngữ truy vấn sử dụng LINQ trên các đối tượng Entity.

Migration: Quản lý các thay đổi trong cơ sở dữ liệu.

Entity & DbContext

Lớp Entity (Thực thể)

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public string Description { get; set; }
}
```

Lớp DbContext

```
using Microsoft.EntityFrameworkCore;

public class MyDbContext : DbContext
{
    public DbSet<Product> Products { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
    }
}
```

LINQ to Entities

```
// Truy vấn với LINQ to Entities
var products = context.Products
               .Where(p ⇒ p.Price > 15)
               .OrderBy(p ⇒ p.Name) .ToList();
foreach (var product in products)
{
        Console.WriteLine($"{product.Name}: {product.Price}");
}
```

Các phương pháp tiến hành EF Core

Database First: Tạo model từ cơ sở dữ liệu có sẵn.

Code First: Viết code để tạo model, sau đó sinh ra cơ sở dữ liệu.

Model First: Thiết kế model qua giao diện, sau đó sinh ra cơ sở dữ liệu.

Các bước thiết lập kết nối

Bước 1. Thêm Entity Framework Core và các gói cần thiết

Sử dụng NuGet Package Manager hoặc lệnh CLI để cài đặt các gói:

```
dotnet add package Microsoft.EntityFrameworkCore
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
dotnet add package Microsoft.EntityFrameworkCore.Tools
```

Nếu sử dụng cơ sở dữ liệu khác (như MySQL, SQLite), thay gói SqlServer bằng gói tương ứng.

Các bước thiết lập kết nối

Bước 2: Kết nối với cơ sở dữ liệu

Đối với Database First:

Sử dụng lệnh CLI để sinh các model từ cơ sở dữ liệu:

```
dotnet ef dbcontext scaffold “ConnStr" Microsoft.EntityFrameworkCore.SqlServer
```

Đối với Code First:

- Tạo các lớp entity và DbContext bằng tay.
- Cấu hình chuỗi kết nối trong file appsettings.json.

Ví dụ

```
private void btnAdd_Click(object sender, EventArgs e)
{
	using (var context = new YourDbContext())
	{
		var newItem = new YourEntity
		{
			Property1 = txtProperty1.Text,
			Property2 = int.Parse(txtProperty2.Text)
		};
		context.YourTable.Add(newItem);
		context.SaveChanges();
		MessageBox.Show("Thêm thành công!");
		LoadData();
	}
}
```

Thực hành