**BÁO CÁO TỔNG KẾT  
DataTable Validation**

MỤC LỤC

[1. Tổng quan 2](#_Toc61713006)

[1.1. Đặt vấn đề: 2](#_Toc61713007)

[1.2. Ý tưởng thực hiện: 2](#_Toc61713008)

[2. Nội dung thực hiện 3](#_Toc61713009)

[2.1. Kỹ thuật FluentValidation xây dựng trình xác thực cho đối tượng 3](#_Toc61713010)

[2.2. Kỹ thuật MathParser xử lý biểu thức toán học 5](#_Toc61713011)

[2.3. Xây dựng chương trình 6](#_Toc61713012)

[2.3.1. Class RuleTable và class Cell 6](#_Toc61713013)

[2.3.2. Class DataProvider 8](#_Toc61713014)

[2.3.3. Class DataTableValidator 11](#_Toc61713015)

[2.4. Giao diện chương trình 12](#_Toc61713016)

[2.4.1. Form Validation 12](#_Toc61713017)

[2.4.2. Xử lý giao diện 14](#_Toc61713018)

[3. Tài liệu tham khảo 15](#_Toc61713019)

# 1. Tổng quan

## 1.1. Đặt vấn đề:

* Bài toán: Xây dựng bộ quy tắc kiểm tra, xác thực dữ liệu cho bảng dữ liệu
* Yêu cầu:
  + Tạo và sử dụng bộ quy tắc xây dựng được để kiểm tra, xác thực dữ liệu cho bảng dữ liệu
  + Đầu vào: Bảng dữ liệu dưới dạng DataTable
  + Đầu ra: Kết quả xác thực, bao gồm danh sách lỗi và thông tin chi tiết về lỗi
  + Code demo, xây dựng giao diện trực quan sinh động

## 1.**2. Ý tưởng thực hiện:**

* Xây dựng cấu trúc quy tắc xác thực dữ liệu (RuleTable) gồm các thành phần:
  + ValidateCell (ô dữ liệu cần xác thực)
  + Operator (toán tử của phép xác thực: >, >=, <, <=, =, …)
  + CompareExpression (biểu thức so sánh)
  + Message (thông báo kết quả xác thực)
* Gộp các quy tắc đơn lẻ lại thành bộ quy tắc (RuleTableCollection), áp dụng cho đối tượng DataTable cụ thể
* Kết hợp sử dụng các thư viện hỗ trợ xử lý tính toán và xác thực dữ liệu
  + Thư viện xác thực dữ liệu FluentValidation
  + Thư viện xử lý toán học MathParser
* Cấu trúc chương trình bao gồm các lớp xử lý, nghiệp vụ:
  + Class **RuleTable**: Định nghĩa cấu trúc của quy tắc xác thực dữ liệu bảng
  + Class **Cell**: Class phụ trợ, định nghĩa cấu trúc ValidateCell trong RuleTable
  + Class **DataProvider**: Kết nối cơ sở dữ liệu, xử lý dữ liệu đầu vào
  + Class **DataTableValidator**: Trình xác thực dữ liệu
  + Form **Validation**: Demo sử dụng bộ quy tắc xác thực dữ liệu

# 2. Nội dung thực hiện

## 2.1. Kỹ thuật FluentValidation xây dựng trình xác thực cho đối tượng

**a) Cơ sở lý thuyết:**

* FluentValidation là thư viện .NET giúp xây dựng các quy tắc xác thực mạnh
* Để xây dựng một tập hợp các quy tắc validation cho một đối tượng, cần tạo một lớp kế thừa AbstractValidator<T>, T loại lớp muốn xác thực.
* Phương thức RuleFor(): Để chỉ định quy tắc xác thực cho một thuộc tính cụ thể. truyền một biểu thức lambda cho biết thuộc tính muốn xác thực.

Ví dụ: Có lớp Customer

**public** **class** **Customer** {

**public** int Id { **get**; **set**; }

**public** string Surname { **get**; **set**; }

**public** string Forename { **get**; **set**; }

**public** decimal Discount { **get**; **set**; }

**public** string Address { **get**; **set**; }

}

* Xác định một tập hợp quy tắc xác thực cho lớp này bằng cách kế thừa từ AbstractValidator<Customer>

Các quy tắc xác thực phải được định nghĩa trong phương thức khởi tạo của lớp trình xác nhận. Ví dụ: lớp CustomerValidator, ràng buộc thuộc tính “Surname” không rỗng.

using FluentValidation;

public class CustomerValidator : AbstractValidator<Customer> {

public CustomerValidator() {

RuleFor(customer => customer.Surname).NotNull();

}

}

* Để chạy trình xác thực, khởi tạo đối tượng trình xác thực và gọi phương thức Validate(), truyền đối tượng vào để xác thực.

Ví dụ :

Customer customer = new Customer();

CustomerValidator validator = new CustomerValidator();

ValidationResult result = validator.Validate(customer);

* Phương thức Validate() Trả về một đối tượng ValidateResult, chứa 2 thuộc tính :
* IsValid - một boolean cho biết việc xác thực có thành công hay không.
* Error - một tập hợp các đối tượng ValidationFailure chứa thông chi tiết về bất kỳ lỗi xác thực nào.

Ví dụ:

Customer customer = **new** Customer();

CustomerValidator validator = **new** CustomerValidator();

ValidationResult results = validator.Validate(customer);

**if**(! results.IsValid) {

**foreach**(var failure **in** results.Errors) {

Console.WriteLine("Property " + failure.PropertyName + " failed validation. Error was: " + failure.ErrorMessage);

}

}

* Có thể gọi ToString() vào ValidationResult để kết hợp tất cả các thông báo lỗi thành một chuỗi duy nhất. Theo mặc định, các thông báo điệp lỗi sẽ được phân tách bằng các dòng mới. Nếu muốn thay đổi tùy chỉnh này, có thể truyền một ký tự phân tách khác cho ToString().

Ví dụ:

ValidationResult results = validator.Validate(customer);

string allMessages = results.ToString("~");

**Xem thêm:** [*https://fluentvalidation.net/*](https://fluentvalidation.net/)

**b) Áp dụng vào chương trình:**

* Xây dựng lớp trình xác thực DataTableValidator, xác thực dữ liệu cho bảng dữ liệu DataTable (xem 2.3.3)

## 2.2. Kỹ thuật MathParser xử lý biểu thức toán học

**a) Cơ sở lý thuyết:**

* MathParser (mXparser) là một [trình phân tích](https://en.wikipedia.org/wiki/Parser) / đánh giá biểu thức toán học [mã nguồn mở](https://en.wikipedia.org/wiki/Open-source_software) cung cấp khả năng tính toán các biểu thức khác nhau tại một [thời điểm chạy](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)). Định nghĩa biểu thức được đưa ra dưới dạng văn bản thuần túy, sau đó được xác minh về [ngữ pháp / cú pháp](https://en.wikipedia.org/wiki/Compiler-compiler) , cuối cùng được tính toán.
* MathParser hỗ trợ:
  + Các tính năng, thành phần có sẵn:
    - Các toán tử cơ bản
    - Quan hệ nhị phân
    - Logic Boolean
    - Các hàm toán học cài sẵn
    - Các hằng số toán học cài sẵn
    - Toán tử được lặp lại
    - Phân biệt và tích hợp số
    - Hỗ trợ số nguyên tố
  + Các thành phần do người dùng định nghĩa:
    - Hằng số do người dùng định nghĩa
    - Đối số do người dùng định nghĩa
    - Các chức năng do người dùng xác định
    - Các hàm đa dạng cho người dùng xác định
    - Đệ quy do người dùng xác định
* Ví dụ:
  + Biểu thức toán học đơn giản và xử lý tính toán

Expression e = new Expression("2+3/(4+5)^4");

double v = e.calculate();

* + Biểu thức toán học với đối số do người dùng định nghĩa và xử lý tính toán:

Argument x = new Argument("x = 5");

Argument y = new Argument("y = 2\*x", x);

Expression e = new Expression("sin(x)+y", x, y);

double v = e.calculate();

**Xem thêm:** [*https://en.wikipedia.org/wiki/Mxparser*](https://en.wikipedia.org/wiki/Mxparser)

**b) Áp dụng vào chương trình:**

* Tạo Expression là CompareExpression trong RuleTable, sử dụng đối số tự định nghĩa để xử lý tính giá trị CompareExpression (xem 2.3.1)

## 2.3. Xây dựng chương trình

### 2.3.1. Class RuleTable và class Cell

**a) Mô tả:**

**Class RuleTable**

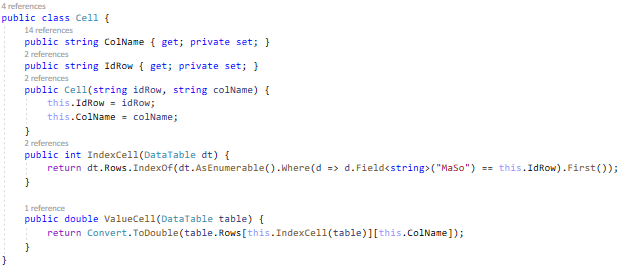
* Định nghĩa cấu trúc một quy tắc xác thực dữ liệu, bao gồm:
  + **DtCell**: Ô dữ liệu cần xác thực (validate)
  + **Operator**: Toán tử của phép xác thực
  + **CompareExpression**: Biểu thức xác thực với dữ liệu của DtCell
  + **ErrorMessage**: Thông báo về lỗi xác thực
* Phương thức khởi tạo **RuleTable**
  + Đầu vào:
    - **strCompareCell:** Chuỗi đại diện cho cell cần validate
    - **strOperator:** Chuỗi cần đại diện cho toán tử của phép xác thực
    - **strCompareExpression:** Chuỗi đại diện cho biểu thức xác thực
    - **listArg:** Danh sách đối số cho xử lý Expression
  + Đầu ra: Đối tượng RuleTable được khởi tạo
* Phương thức **ToCompareCell**
  + Đầu vào:
    - **strCompareCell:** Chuỗi đại diện cho cell cần validate
  + Đầu ra: Thành phần DtCell của đối tượng RuleTable
* Phương thức **ToCompareExpressionValue**
  + Đầu vào:
    - **strCompareExpression:** Chuỗi đại diện cho biểu thức so sánh
    - **listArg:** Danh sách đối số cho xử lý Expression
  + Đầu ra: Giá trị thành phần CompareExpression của đối tượng RuleTable

**Class RuleTable** (class phụ trợ cho class RuleTable)

* Định nghĩa cấu trúc ValidateCell trong RuleTable, bao gồm:
  + **ColName:** Tên cột của cell
  + **IdRow:** Mã dòng của cell
* Phương thức khởi tạo **Cell:** Khởi tạo DtCell cho RuleTable từ chuỗi đại diện
* Phương thức **IndexCell:** Lấy về chỉ số hàng của cell trong DataTable
* Phương thức **ValueCell:** Lấy về giá trị của cell trong DataTable

**b) SourceCode:**

****

****

### 2.3.2. Class DataProvider

**a) Mô tả:**

* Chức năng: Kết nối cơ sở dữ liệu, xử lý dữ liệu đầu vào
* Thuộc tính:
  + **conStr:** Chuỗi kết nối tới cơ sở dữ liệu (database)
  + **dt:** DataTable trung gian, lưu DataTable chuẩn bị cho validate
  + **listArg:** HashSet lưu các Argument trong Expression đọc từ database
* Phương thức **GetListArgument:**
  + Đầu vào:
    - **reportCode:** Mã báo cáo của bảng dữ liệu cần validate
    - **table:** DataTable lưu dữ liệu cần validate
  + Nội dung xử lý:
    - Kết nối database lưu RuleTable, đọc ra các Expression từ các RuleTable có ReportCode = **reportCode** (lọc theo mã báo cáo)
    - Dùng StringBuilder cộng các Expression thành Expression tổng hợp
    - Dùng phương thức **getMissingUserDefinedArgument (**của Mxparser) để tách và lấy các Argument trong Expression tổng hợp

(getMissingUserDefinedArgument là phương thức của lớp Expression trong Mxparser, có chức năng nhận dạng và phân tách các Argument trong Expression, trả về 1 mảng kiểu string[] )

* + - Dùng phương thức **ToDictionaryValue** (tự định nghĩa) để mapping KeyValue, đưa vào Dictionary

(Key: argument(lấy từ trên, đại diện 1 cell), Value: giá trị của cell đó)

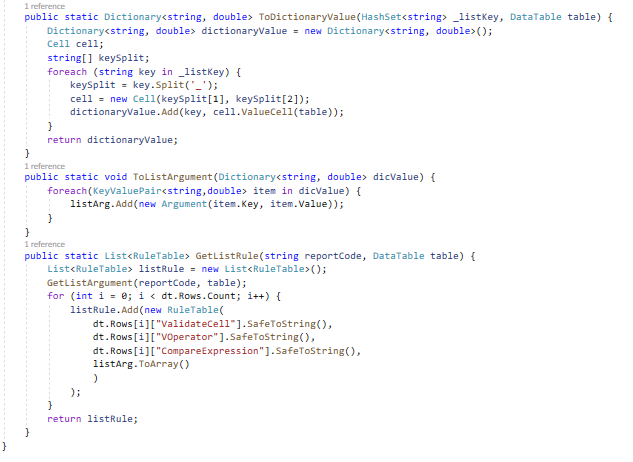
* + - Dùng phương thức **ToListArgument** (tự định nghĩa) để lấy ra danh sách Argument, lưu vào biến toàn cục listArg
  + Kết quả: Danh sách Argument lọc ra được từ các Expression trong database, lưu vào biến toàn cục listArg
* Phương thức **ToDictionaryValue:**
  + Đầu vào:
    - **\_listKey:** danh sách các Argument (sử dụng làm Key cho DictionaryItem)
    - **table:** DataTable lưu bảng dữ liệu cần validate
  + Nội dung xử lý:
    - Khởi tạo 1 Dictionary
    - Duyệt \_listKey, lấy ra từng phần tử là Argument đại diện cho cell
    - Tách chuỗi để lấy ra IdRow và ColName của cell, khởi tạo 1 Cell
    - Dùng phương thức **ValueCell** (class Cell) để tính giá trị cell
    - Thêm cặp Key-Value (argument đại diện cell, giá trị cell) vào Dictionary
  + Đầu ra: Dictionary <Argument đại diện cell, giá trị của cell>
* Phương thức **ToListArgument:**
  + Đầu vào:
    - **dicValue:** Dictionary <Argument đại diện cell, giá trị của cell>
  + Nội dung xử lý:
    - Duyệt Dictionary
    - Lấy cặp Key-Value của mỗi phần tử để khởi tạo Argument mới

(ArgumentName = Key, ArgumentValue = Value)

* + - Thêm Argument mới vào listArg toàn cục
  + Kết quả: Danh sách Argument, lưu vào biến toàn cục listArg
* Phương thức **GetListRule:**
  + Đầu vào:
    - **reportCode:** Mã báo cáo của bảng dữ liệu cần validate
    - **table:** DataTable lưu bảng dữ liệu cần validate
  + Nội dung xử lý:
    - Khởi tạo List<RuleTable> mới
    - Dùng phương thức **GetListArgument** (tự định nghĩa) để nạp dữ liệu danh sách Argument vào biến toàn cục listArg
    - Duyệt các dòng trong DataTable, khởi tạo RuleTable tương ứng, thêm vào List<RuleTable>
  + Đầu ra:
    - List<RuleTable>: Danh sách quy tắc xác thực cho một bảng dữ liệu báo cáo với mã báo cáo cụ thể

**b) SourceCode:**

****

****

### 2.3.3. Class DataTableValidator

**a) Mô tả:**

* Sử dụng kỹ thuật FluentValidation (2.1) để xây dựng trình xác thực cho DataTable
* Đầu vào:
  + DataTable: DataTale lưu dữ liệu bảng báo cáo cần validate
  + List<RuleTable>: Danh sách quy tắc xác thực cho DataTable
* Kết quả:
  + Kết quả xác thực dữ liệu

**b) SourceCode:**



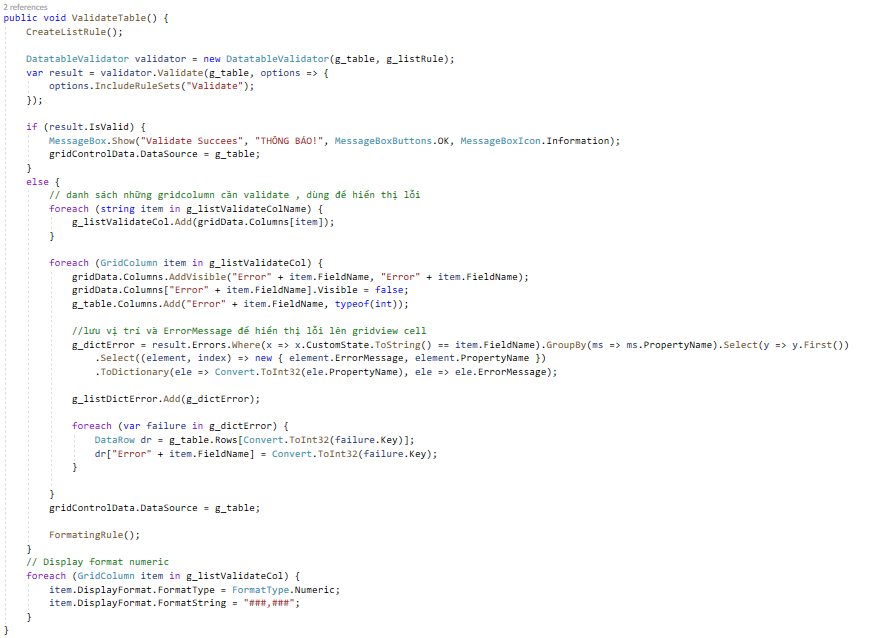
## 2.4. Giao diện chương trình

### 2.4.1. Form Validation

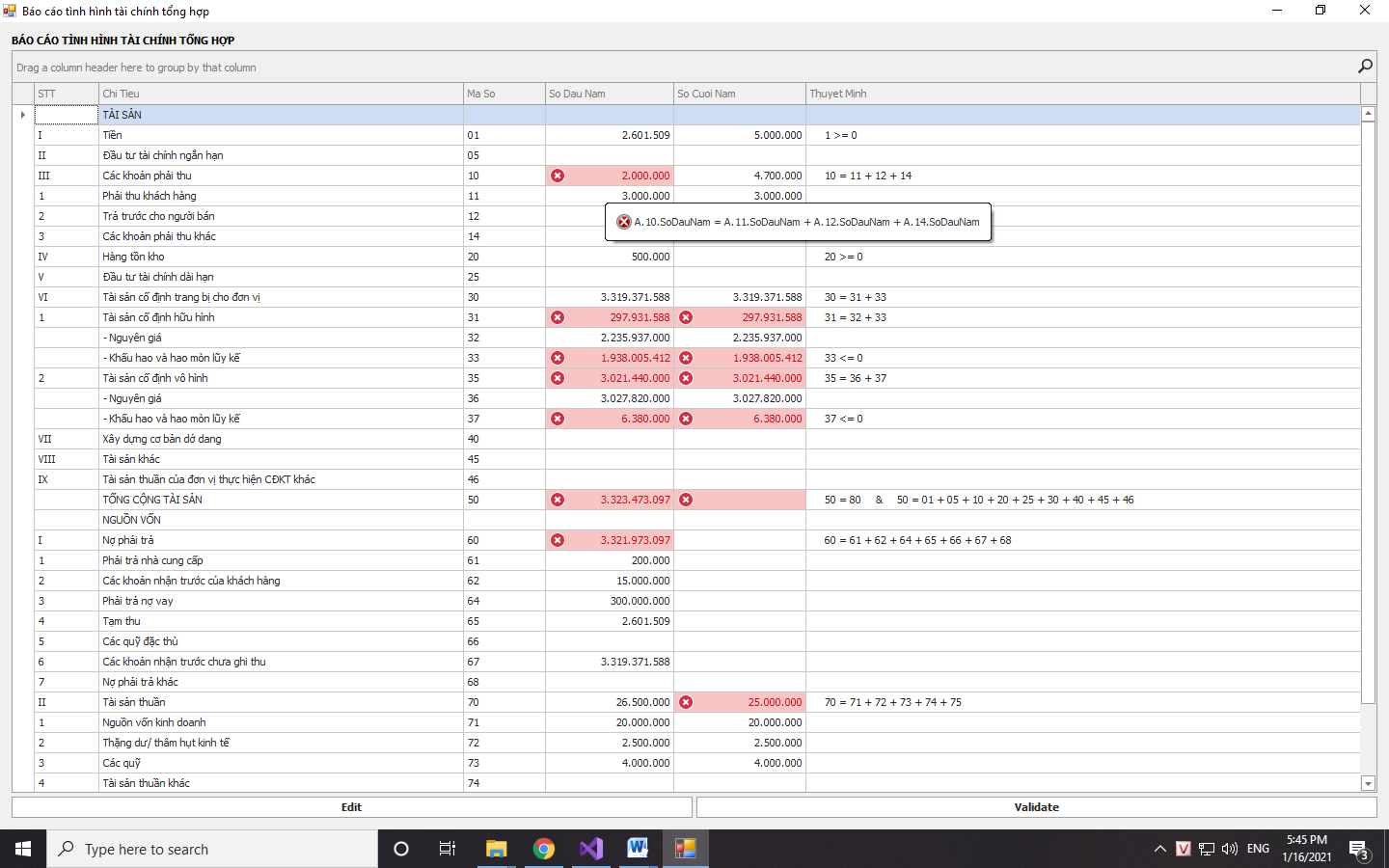
* Khởi tạo List<RuleTable>



* Thực hiện validate DataTable



* Kết quả Validate



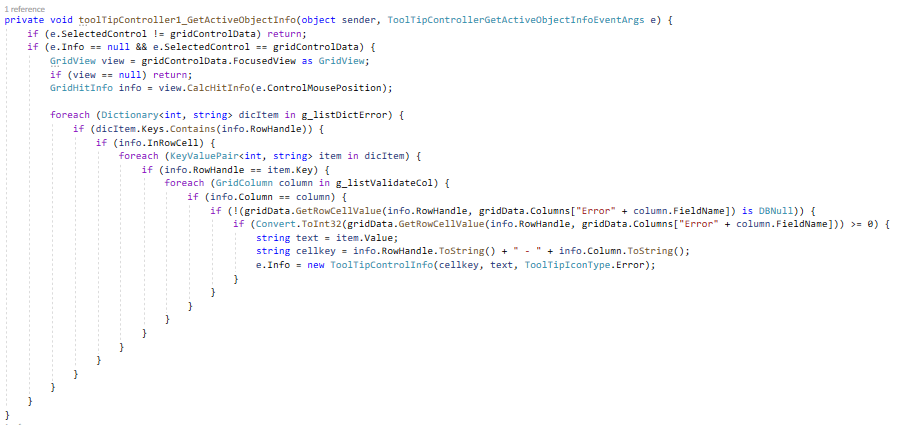
* Những cell có dữ liệu không đúng theo điều kiện được hightlight kèm theo biểu tượng lỗi
* Trỏ chuột vào cell bị lỗi, có thông báo hiện lên về chi tiết lỗi tại cell đó
* Bấm chọn nút “Edit” để chỉnh sửa dữ liệu tại các cell
* Bấm chọn nút “Validate” để thực hiện validate lại dữ liệu trong bảng
* Full SourceCode xử lý trong FormValidation (xem project đính kèm báo cáo)

### 2.4.2. Xử lý giao diện

**a) Hightlight nội dung lỗi (sử dụng FormatingRule)**

****

**b) Sử dụng ToolTipControl để hiển thị thông tin chi tiết lỗi khi trỏ chuột**

****

# 3. Tài liệu tham khảo

Link trang web:

* <https://fluentvalidation.net/>
* <https://supportcenter.devexpress.com/ticket/list>
* <https://en.wikipedia.org/wiki/Mxparser>
* <https://www.codeproject.com/Tips/381509/Math-Parser-NET-Csharp>
* <https://www.mathparsers.com/math-parser-for-csharp-dotnet/>