# Code Convension .NET

## Tabs & Indenting

Tab characters (\0x09) should not be used in code. All indentation should be done with 4 space characters.

## Bracing

Open braces should always be at the beginning of the line after the statement that begins the block. Contents of the brace should be indented by 4 spaces. For example:

|  |
| --- |
| if (someExpression)  {  DoSomething();  }  else  {  DoSomethingElse();  } |

“case” statements should be indented from the switch statement like this:

|  |
| --- |
| switch (someExpression)  {  case 0:  DoSomething();  break;  case 1:  DoSomethingElse();  break;  case 2:  {  int n = 1;  DoAnotherThing(n);  }  break;  } |

Braces should never be considered optional. Even for single statement blocks, you should always use braces. This increases code readability and maintainability.

**Parentheses should be used in expressions that involve mixed operators:**

if ~~(x>0 && x<3)~~ => if ((x>0) && (x<3))

|  |
| --- |
| for (int i = 0; i < 100; i++) { DoSomething(i); } |

## Single line statements

Single line statements can have braces that begin and end on the same line.

|  |
| --- |
| public class Foo  {  int bar;  public int Bar  {  get { return bar; }  set { bar = value; }  }  } |

It is suggested that all control structures (if, while, for, etc.) use braces, but it is not required.

## Commenting

Comments should be used to describe intention, algorithmic overview, and/or logical flow. It would be ideal, if from reading the comments alone, someone other than the author could understand a function’s intended behavior and general operation. While there are no minimum comment requirements and certainly some very small routines need no commenting at all, it is hoped that most routines will have comments reflecting the programmer’s intent and approach.

## Copyright notice

Each file should start with a copyright notice. To avoid errors in doc comment builds, you don’t want to use triple-slash doc comments, but using XML makes the comments easy to replace in the future. Final text will vary by product (you should contact legal for the exact text), but should be similar to:

|  |
| --- |
| //------------------------------------------------------------------  // <copyright file="ContainerControl.cs" company= VNYI">  // Copyright (c) VNYI. All rights reserved.  // </copyright>  //------------------------------------------------------------------ |

## Documentation Comments

All methods should use XML doc comments. For internal dev comments, the <devdoc> tag should be used.

|  |
| --- |
| public class Foo  {  /// <summary>Public stuff about the method</summary>  /// <param name=”bar”>What a neat parameter!</param>  /// <devdoc>Cool internal stuff!</devdoc>  ///  public void MyMethod(int bar) { … }  } |

However, it is common that you would want to move the XML documentation to an external file – for that, use the **<include>** tag.

|  |
| --- |
| public class Foo  {  /// <include file='doc\Foo.uex' path='docs/doc[@for="Foo.MyMethod"]/\*' />  ///  public void MyMethod(int bar) { … }  } |

## Comment Style

The // (two slashes) style of comment tags should be used in most situations. Where ever possible, place comments above the code instead of beside it. Here are some examples:

|  |
| --- |
| // This is required for WebClient to work through the proxy  GlobalProxySelection.Select = new WebProxy("http://itgproxy"); |

And

|  |
| --- |
| // Create object to access Internet resources  //  WebClient myClient = new WebClient(); |

Comments can be placed at the end of a line when space allows:

|  |
| --- |
| public class SomethingUseful  {  private int itemHash; // instance member  private static bool hasDoneSomething; // static member  } |

## Spacing

Spaces improve readability by decreasing code density. Here are some guidelines for the use of space characters within code:

Do use a single space after a comma between function arguments.

**Right:** Console.In.Read(myChar, 0, 1);

Wrong: Console.In.Read(myChar,0,1);

* Do not use a space after the parenthesis and function arguments

**Right:** CreateFoo(myChar, 0, 1)

Wrong: CreateFoo( myChar, 0, 1 )

* Do not use spaces between a function name and parenthesis.

**Right:** CreateFoo()

Wrong: CreateFoo ()

* Do not use spaces inside brackets.

**Right:** x = dataArray[index];

Wrong: x = dataArray[ index ];

* Do use a single space before flow control statements

**Right:** while (x == y)

Wrong: while(x==y)

* Do use a single space before and after comparison operators

**Right:** if (x == y)

Wrong: if (x==y)

## Naming

**classes:**   
      CapitalizedWithInternalWordsAlsoCapitalized   
**methods:**   
      firstWordLowerCaseButInternalWordsCapitalized   
      start the method name with a verb   
**private or protected methods:**   
      \_firstWordLowerCaseButInternalWordsCapitalized

**constants (finals):**   
      UPPER\_CASE\_WITH\_UNDERSCORES   
**string constants (finals):**   
      S\_UPPER\_CASE\_WITH\_UNDERSCORES   
**float/double constants (finals):**   
      F/D\_UPPER\_CASE\_WITH\_UNDERSCORES   
**private or protected class variable:**   
      \_leadingUnderscore

**properties or public variable:**

      LikePrivateButFirstLetterCapitalized   
**method local variables:**   
      firstWordLowerCaseButInternalWordsCapitalized   
**method parameter variable:**   
      trailingUnderscore\_

**Do prefix interfaces names with “I”**

## File Organization

Source files should contain only one public type, although multiple internal classes are allowed

Source files should be given the name of the public class in the file

Directory names should follow the namespace for the class

For example, I would expect to find the public class “System.Windows.Forms.Control” in “System\Windows\Forms\Control.cs”…

Classes member should be alphabetized, and grouped into sections (Fields, Constructors, Properties, Events, Methods, Private interface implementations, Nested types)

Using statements should be inside the namespace declaration.

|  |
| --- |
| using System;  namespace MyNamespace  {  public class MyClass : IFoo  {  // fields  int foo;  // constructors  public MyClass() { … }  // properties  public int Foo { get { … } set { … } }  // events  public event EventHandler FooChanged  {  add { … } remove { … }  }  // methods  void DoSomething() { … }  void FindSomethind() { … }  //private interface implementations  void IFoo.DoSomething() { DoSomething(); }  // nested types  class NestedType { … }  }  } |

|  |  |  |
| --- | --- | --- |
| **No** | **TYPE** | **ABB.** |
| **Primitive type(optional)** | | |
|  | **byte** | by |
|  | **short** | t |
|  | **int** | i |
|  | **long** | l |
|  | **float** | f |
|  | **double** | d |
|  | **char** | c |
|  | **boolean** | b |
|  | **String** | s |
|  | **StringBuffer** | sb |
| **User Interface** | | |
|  | **Button** | btn |
|  | **BindingNavigator** | bnav |
|  | **BindingSource** | bs |
|  | **Checkbox** | ck |
|  | **CheckedListBox** | clb |
|  | **Class** | cls |
|  | **Combobox** | cbo |
|  | **DataGridView** | grd |
|  | **DataSet** | ds |
|  | **Dialog** | dlg |
|  | **Exception** | ex |
|  | **Form** | frm |
|  | **Label** | lb |
|  | **List** | lst |
|  | **Menu** | mn |
|  | **Menu Item** | mn |
|  | **Menubar** | mnb |
|  | **Object** | obj |
|  | **Panel** | pn |
|  | **Popup Menu** | mnp |
|  | **Progress Bar** | prg |
|  | **Radio Button** | rdo |
|  | **Radio Button Menu Item** | mnr |
|  | **Scroll Pane** | scp |
|  | **Scrollbar** | scb |
|  | **Separator** | sep |
|  | **Slider** | sld |
|  | **Spinbox** | spn |
|  | **SplitPane** | spp |
|  | **StatusStrip** | sta |
|  | **Tabbed Pane** | tb |
|  | **Text Area** | txa |
|  | **Text Field** | txt |
|  | **Text Pane** | txp |
|  | **Timer** | tmr |
|  | **Toggle Button** | tog |
|  | **Tool Bar** | tbr |
|  | **Tool Tip** | tip |
|  | **Toolbar Button** | tbt |
|  | **Repater** | rep |
|  | **Div** | dv |
|  | **LinkButton** | lk |
|  | **Static** | \_ |
|  | **hiddenField** | hid |
|  | **Store** | sp |
|  | **View** | vw |
|  | **Function** | fn |
|  | **Namespace** | vnyi. |
|  | **GridPanel Column** | cl  vd: cột mã : clID, Tên :clName |



**Diễn giài :**

GridView (dll)

Dll chứa lưới xử lý chung

**Accountin (Page)**

Tất cả page, usercontrol, javascript,css được thiết lập thành các folders theo các modul theo các chức năng riêng biệt .

* **Usercontrols** là folder chứa các usercontrol chung
* **SaleOrder** chứa các trang về bán hàng
* **Purchasing** folders chứa trang mua hang
* **AR,AP,CM,IC,CM** chứa các trang về nghiệp vụ tương ứng
* **CTM** Contract Managerment
* **Finder** chứa page finder
* **Javascript** folder chứa các file js
* **Images** folders chứa các icon,hình anh ..
* **ASM** Asset Management trang về tài sản
* **Css** chứa các css

**DataProvider (dll)**

Là tầng kết nối với database và trả về dữ liệu thô là một dataset hoặc một kiểu int tuy vào trường hợp cụ thể trước hết là lấy cấu hình và lấy chuỗi kết nối từ database access, lấy chuỗi kết nối theo cấu trúc của tổ trức và kết nối để lấy data từ database tương ứng . tầng này sử dụng **Enterprice Library** và một số thư viện có sẵn

**GridViewCustomFields (dll)**

Là thu viện chứa lưới được xây dựng lại theo **Ezipos**

Library (dll)

Tầng này chứa tất cả **object** theo mô hình hướng đối tượng, được xây dựng theo từng folder và theo từng Namespace. Một **table** dưới **database** tương ứng sẽ có hai class, class mang tên table là class đối tượng chứa tất cả thuộc tính tương ứng với đối tượng đó, class mang tên **table** và có đuôi là **DAO** là class chứa các method nhận một sự kiện o trên tẩng Accounting . Lấy dữ liệu thô lên từ **DataProvider** và convert thành data có cấu trúc , trả về là một **Data** cấu trúc theo chuẩn hướng đối tượng . Tầng này chứa các method về data

* MES object về thông báo
* META object về MetaData
  + ACC object Accountting
  + CONTROL object chứa các menu và control trên trang
  + Extent đối tượng mở rộng như ngôn ngữ ...
  + Item object về hang hóa, kho hang …..
  + OBJ object về một đối tượng nhà cung cấp, nhân viên, contact …
  + ORG object về tổ trức
  + STATUS object về trạng thái
* PLAN object về kế hoạch
* REQ object về yêu cầu
* PO object về Purchase order(bán hang)
* SHIP object về vận chuyển
* SO object về sale order
* User object về user

Workflow (dll)

Tầng này chứa một cấu trúc theo Workflow đã được dựng sẵn theo cấu trúc cố định để quản lý trạng thái của các document theo luồng nghiệp vụ

Utility(dll)

Chứa các hàm mở rộng về format về datetime … và chứa một số method dung chung ..

**Quy dinh code nghiep vu**

## Mô hình Codding :

Với mô hình master detail thì nghiệp vụ và chi tiết nghiệp vụ được lưu xuống một bảng tmp khi lưu thì chỉ mang thông tin từ bảng tmp add vào bảng chính và sử dụng transaction tại đây

Thông báo lỗi :

Phong đã xây dựng một class Msg thông báo về lỗi của một nghiệp vụ hay một lưới.

Trước tiên muốn sử dụng cảnh báo này tại server chúng ta tạo ra một Msg thực tế Msg là một danh sách thông báo có thể là thông báo cho các control và có thể thông báo trên lưới

**Tạo thông báo control**

Msg msg = new Msg();

Và Phải using vnyi.Utility;

Tạo thông báo control

msg.add("ControlName", "Messager");

ControlName : Cho biế ID của control

Messager: Thông báo khi control bi loi

**Tạo thông báo tren lưới**

msg.add("GridPanelID", RowIndext, "CUR\_NAME", "Massager");

GridPanelID : Id của lưới cần thông báo

RowIndext : Dòng cần thông báo

CUR\_NAME : tên cột muốn hiển thị thông báo

Massager: Câu thông báo

msg.add Thêm thông báo vào danh sách

Tại server trả về danh sách lỗi

protected void btnSave\_Click(object sender, AjaxEventArgs e)

{

e.Success = false;

e.Success = false; báo cho client biết là sự kiện này bị lỗi

tạo thông báo

Msg msg = new Msg();

msg.add("txtName", "dai qua");

msg.add("GridPanel1", 1, "CUR\_NAME", "nhap vao");

gán thông báo cho AjaxEventArgs Parametter ErrorMessage

e.ErrorMessage = JSON.Serialize(msg);

JSON.Serialize dùng convert qua Json Từ một **list<Generic>**

Khi trả về là một chuỗi theo cấu trúc của một Json .

Phía trên client Trong sự kiện của button

<AjaxEvents>

<Click OnEvent="btnSave\_Click"

Scope="this"

Before = "validation();"

Success="onSuccess();"

Failure="onFailure(response, result, el, type, action, extraParams);" >

<ExtraParams>

<ext:Parameter Name="valuesAll" Value="Ext.encode(#{GridPanel1}.getRowsValues(false))" Mode="Raw" />

<ext:Parameter Name="valuesChange" Value="#{GridPanel1}.getView().ds.getChangedData()" Mode="Raw" />

</ExtraParams>

</Click>

</AjaxEvents>

btnSave\_Click La một sự kiện ajax ở server

Before = "validation();" dùng hàm validation ở client trước khi một sự kiện ajax xảy ra ta có thể dùng hàm này check một số validate ở client

Success="onSuccess();" hàm này dược viết ở client khi sự kiện ajax được hoàn thành

Failure="onFailure(response, result, el, type, action, extraParams);" > hàm này viết ở client xảy ra khi sự kiện ajax được báo là false với response là một đối tượng trả về từ server và hàm **onFailure được viết tại file ErrorMessage.js** hàm này có nhiệm vụ thông báo lỗi cho lưới chi gọi ra sử dụng

Trên trang aspx muốn sử dụng báo lỗi này phải viết thêm hai hàm

// Thong bao thanh cong

function onSuccess() {

}

// Kiem tra du lieu tai client

function validation() {

resetValidate();

}

resetValidate dùng để Clear những thông báo trước đó

ứng với nghiệp vụ có master và nhiều detail thì khi detail trong sự kiện thêm mới khi thêm detail thì lúc này Lưu bussiness xuống bảng tmp của nó và trả về ID trong bảng tmp Id này được gán vào detail khi xuống dòng tại detai thì Insert dòng này tới bảng tmp của detail tương ứng của nó

khi nhấn nút Save tren toollbar thì lúc này ra một lệnh về server ra lệnh coppy từ bảng tmp vào bảng chính và sử dụng transaction tại đây

trường hợp cập nhật Bussiness khi do sẽ lấy data ở bảng chính đưa vào bảng tmp mọi sử lý thêm xóa sửa được thao tác trên bảng tmp khi nhấn nút lưu thì xóa data ở bảng chính và lấy thông tin trên bảng tmp coppy vào bảng chính và sử dụng transaction

**CÁCH XỬ LÝ LƯU DỮ LIỆU VÀO DATABASE**

## QUY ĐỊNH THIẾT LẬP MENU CHUẨN CHO CÁC FORM TRÊN ERP

## Menu cho Form nghiệp vụ

form nghiep vu

1. Tạo mới: PageAdd
2. Sửa: PageEdit
3. Lưu: Disk
4. Bỏ qua: Cancel
5. Xóa: Delete
6. In: Printer
7. Các button Navigator |< < > >|
8. Trợ giúp : Help
9. Đóng : Stop
10. Chức năng: Icon cid:image002.png@01CB06EE.8CE7B690 CogEdit
11. Vận chuyển: Icon 
12. Tìm Trên các textbox để gọi finder: Icon 

* Các button Trợ giúp, Đóng nằm ở góc phải màn hình

## Menu cho form Danh sách nghiệp vụ (GL List, CM List, IC List …)

form danh sach nghiep vu

1. Tạo mới: pageadd
2. Sửa
3. Xóa
4. Tìm kiếm : find
5. In : print
6. Đóng : stop
7. Xem: dung Icon: Application ViewIcon
8. Ghi sổ: Icon Ghi sổ:  BookAdd=> Sổ chinh, BookAdressesAdd: Icon 🡺 Sổ Tạm
9. Xuất dữ liệu: Icon  DatabaseGo
10. Trợ giúp: Help
11. Đóng : stop

## Menu cho form Danh sách Danh mục (Currency List, TAX List …)

form danh sach danh muc

1. Tạo mới
2. Sửa
3. Xóa
4. Làm mới : ArrowRefresh / Tìm kiếm : Find
5. In : print
6. Nhập dữ liệu: DatabaseAdd
7. Xuất dữ liệu: Icon  DatabaseGo
8. Trợ giúp : help
9. Đóng : stop

Button “Tìm kiếm” dùng trong các form nhiều dữ liệu, có các combo, textbox để nhập dữ liệu tìm kiếm ( danh mục hàng hóa, đối tượng)

Button “Làm mới” dùng trong các form có dữ liệu ít ( danh mục tiền tệ,..) . Chỉ dùng button “Tìm kiếm” hoặc “Làm mới” , không dùng cả 2 button trên 1 form

## Menu cho form Thông tin Danh mục (Currency, TAX, USER …)

form danh muc

1. Tạo mới
2. Lưu
3. Bỏ qua
4. Xóa
5. Các button Navigator |< < > >|

## Menu cho Finder

menufinder

1. Chấp nhận : accept
2. Bỏ qua : cancel
3. Tìm kiếm: Find

## Qui định khác

* Thể hiện trường dữ liệu bắt buộc có giá trị nhập là “\*” màu đỏ
* Khi tạo mới hoạc sửa, đóng form phải xác nhận lại với User.

## Quy định màu trạng thái.

## QUY ĐỊNH ĐẶT TÊN STORE VÀ GHI CHÚ, DIỄN GIẢI CÁC DÒNG CODE TRONG STORE

* **Cách Comment nội dung store**

Đầu mỗi Store cần ghi nhận thông tin như bên du7oi1. Đối với các store do hệ thống tạo ra thi để tên là “System”.

/\* ======================================================

Author: Name

Create date: Date

Description: Description

Modified date: Date

Modified By: Name — write purpose modified

========================================================

\*/

## Cần comment đối với các block code của store, với nội dung ghi ra rằng viêc thực thi của block này nhằm 1 mục đích gì. Chú ý , đối với các block thì nên view theo layout chuẩn dung Red Gate để format.

/\* --- Comment block --- \*/.

* **Cách đặt tên cho Store mình viết.**

**Cách đặt tên store theo nguyên tắc như sau:**

***sp\_Phân hệ\_Chức năng\_Tên bang hoặc đối tượng cần thực hiện***

**VD: sp\_IC\_GetItemDetails\_ICInstockDetail (Store này lấy danh sách chi tiết nhập kho hang hóa)**

***Ghi chú:***

Sp: là ký hiệu đánh dấu cho biết đây là store procedure

Phân hệ: Là đánh dấu store này viết cho phan hệ nào. Nếu store viết dung chung thì dung từ “PUB”. Nếu dùng cho report thì phần này gán thêm chu “report”. VD: sp\_ARReport\_

Chức năng: Thể hiện hành động của store này là gì. Ví dụ: select, get, insert, update, Search…

Tên bang hoặc đối tượng cân thực hiện: Thể hiện tên bang hoac đối tượng thực hiện lien quan nao đó. Nếu có nhiều đối tượng hoặc nhiểu bảng chung thi có thể đặt tên nào đó gợi ý cho nghiệp vụ của store đó.

## Quy định comment code.

Tại cac class tuân thủa commend class về thong tin class, muc đich su dung class va 1 so bien khai báo trong class cần ghi chú.

public class Foo

{

/// <summary>Public stuff about the method</summary>

/// <param name=”bar”>What a neat parameter!</param>

/// <devdoc>Cool internal stuff!</devdoc>

///

public void MyMethod(int bar) { … }

}

|  |  |  |  |
| --- | --- | --- | --- |
| **Một số control dùng chung** | | | |
| **Loại** | **ID** | **Text** | **Icon** |
| **Button thêm mới** | btnNew | Tạo mới | PageAdd |
| **Button cập nhật** | btnUpdate | Sửa | PageEdit |
| **Button Xóa** | btnDelete | Xóa | Delete |
| **Button Xem** | btnView | Xem | ApplicationViewDetail |
| **Form detail(có UserControl)** | | | |
| **Button thêm mới** | btnNew | Tạo mới | PageAdd |
| **Button cập nhật( nếu có)** | btnUpdate | Sửa | PageEdit |
| **Button Xóa** | btnDelete | Xóa | Delete |
| **Control dưới lưới chi tiết của form master** | | | |
| **Button thêm dong** | btnNewline | Thêm dòng |  |
| **Button xóa dong** | btnDeleteLine | Xóa dòng |  |
| **Control trên form detail( không dùng usercontrol)** | | | |
| **Thêm mới** | btnNewDetail | Tạo mới | PageAdd |
| **Cập nhật** | btnUpdateDetail | Sửa | PageEdit |
| **Xóa** | btnDeleteDetail | Xóa | Delete |
| **Xem** | btnView | Xem | ApplicationViewDetail |

1. **Tạo chức năng Duyệt:** Làm 2 button **“Duyệt”** và **“Hủy duyệt”.**

Tên của button đặt tên là: btnApprove, btnUnApprove. Cho phép chuyển active với 2 btn này chỉ có 1 btn được active tại 1 thời điểm.

|  |  |  |  |
| --- | --- | --- | --- |
| **Duyệt** | btnApprove | Duyệt | TransmitGo |
| **Hủy Duyệt** | btnUnApprove | Hủy duyêt | Decline |

* Trang thái đã duyệt thì không được phép sửa mà chỉ view.
* Thông tin sẽ được sửa khi danh mục hoặc document đó có trạng thải là Hủy duyệt.
* Trường hợp khi form có nút **Lưu** và **Sửa** thì với trạng thái Duyệt hệ thống không cho phép active click thực hiện các chức năng này
* Tương tự đối với nút **Xóa**: Trạng thái duyệt không cho phép thực hiện xóa mà chỉ có thể xóa khi bỏ duyệt.
* Khi đang sửa mà nhấn vào nút Duyệt nhưng lúc đó chưa lưu thông tin thì hệ thống chỉ duyệt những thông tin cũ. Nhưng thông tin mới do chưa lưu nên không thể hiện sau khi duyệt.
* Khi “tạo mới” không thể hiện button duyệt và hủy duyệt, chỉ thể hiện khi “cập nhật”

## Trạng thái cập nhật trong form danh mục

Đối với các form danh mục trạng thái Thêm mới và xóa thì đã thống nhất giống như của Chứng từ Document.

* Trường hợp cập nhật. khi show lên form detail cho phép duyệt và xem thông tin detail. Nếu có sửa thông tin trên Entry tuy nhiên khi click vào nút Lưu mà Không có quyền “Sửa ” thì thông báo lên “Bạn không có quyền sửa thông tin này.”

Hiện tại các form danh mục vẫn ok. Theo nguyên tác như Chỉnh đã trển khai chúng ta sẽ làm cho phần phân quyền.

## Quy định Tooltip trên form

Các Button trên menu cả danh mục hoặc nghiệp vụ nếu có gắn phím tắc thì phải show tooltip.

Chử tooltip được In đậm và thể hiện khi Drag chuột ngang Button đó.

## Quy định trạng thái của các Button khi thao tác trên Danh mục.

* **Thêm mới**: các button khac thể hiện như sau
  + Check quyền **Thêm mới**
  + Button Tạo mới: Mờ đi, khi nhấn “ESC” hoặc “Bỏ qua” thì quay lại load vào Records dang focus trên lưới.
  + Lưu: Hiện lên để sau khi thêm mới thì click vao “Lưu” để lưu.
  + Xóa: Mờ đi vì tác vụ này không lien quan đến nghiệp vụ xóa.
  + In: Mờ đi khi thêm mới chưa lưu hoặc lưu không thành công.
  + Các Button Navigator không được phép active.
  + Bỏ qua: Nếu form có nút bỏ qua thì nút bỏ qua hiện lên
  + Nhập, xuất dữ liệu: Nếu form dạng lưới và cho phép nhập thể hiện thông tin trên panel bên trái thì khi thêm mới các Button “Nhập, Xuất dữ liệu ” mờ đi
  + Nút Trợ giúp và dóng vẫn thể hiện
  + Đối với trường hợp không có thể hiện form chi tiết
* **Xóa dữ liệu**
  + Check quyền **Xóa**
  + Hiện lên câu thông báo xác nhận thông tin xóa
    - Đồng ý thì tiến hành chức năng xóa
    - Không thì bỏ qua chức năng này và quay lại trạng thái ban đầu View lại thông tin của dòng dữ liệu đang focus trên lưới.
* **Sửa Danh mục**
  + **Đối với form không có form chi tiết.**
    - Thông tin khi thể hiện trên lưới và load dữ liệu dòng đang Focus nếu muốn sửa thì thay đổi thông tin của thực thể đó Trên Panel Trái 🡺 Click vào Nút “Lưu” để lưu sự thay đổi 🡺 Kiểm quyền “Lưu” Nếu có quyền thì cho lưu, nếu không thì hiện lên thông báo Bạn không có quyền sửa thông tin.
    - Khi đồng ý thì thể hiện lại thông tin Records dang focus của lưới.
  + **Đối với Form có Chi tiết** 
    - Từ danh mục Click vào Button “Xem”. các Button có text “Sửa” thì chuyển lại là “Xem”. Hoặc Double click để mở form.
    - Hiện lên form chi tiết để sửa thông tin
    - Tiến hành sửa thông tin
    - Click vào nút “Lưu” để lưu sự thay đổi. 🡺 check quyền “sửa”. Nếu có quyền thì thực hiện Update thông tin còn không có thì thể hiên câu thông báo Không có quyền sửa thông tin.

1. **Xử lý check quyền trên form**

Nút “View”🡺 “Xem” chúng ta dung Icon: Application View Detail

* 1. **Form danh mục có form Detail**
     1. Trong form danh sách sửa lại Nút “sửa” thành “Xem”
     2. Trên form danh mục có 3 control: chuẩn btnNew,btnUpdate,btnDelete (Nếu có chức năng In hoac Xuất, nhập File thì Check Cntrol “btnPrint”, “btnExport”, “btnImport”). Các control phân quyền này nằm trên form danh sách.
     3. Thêm mới: Check quyền “btnNew”: Khi click thêm mới thì sẽ check quyền trước. Khi thêm click vào Button New tại form danh sách hoặc Detail. 🡺 Sau đó click vào nút “Lưu” trên form Detail để lưu (Lúc này trang thái Record là tao mới nên nút lưu ko cần check quyền)
     4. Sửa: Check quyền “btnUpdate”: button này được ẩn.
        1. Khi muốn sửa thì Double Click dể view Detail hoac kích vào nút “Xem”. =🡺 Sửa thông tin 🡺 Click và nút “Lưu lại”. Lúc này sẽ check quyền của control “btnUpdate”. Nếu không có quyền thì thông báo.
     5. Xóa thì check quyền control “btnDelete” để xác nhận quyền.
  2. **Form danh mục không có form Detail (Panel detail bên trái của form List)**

Các control cũng có tên tương tự như các form danh mục có detail. Chi khác khi sửa hoăc tạo mới thì không có view form detail.

* Thêm mới: Check quyền “btnNew”: Khi click thêm mới thì sẽ check quyền trước khi tạo mới ô nhập để nhập thông tin mới
* Sửa: Check quyền ”btnUpdate”. Button này được ẩn. Khi sửa thông tin =🡺 Lưu thì check quyền Control này.
* Xóa thì check quyền control “btnDelete” để xác nhận quyền
  1. **Form Chứng từ:** danh sách các control phân quyền vẫn nằm trên form Danh sách. Các button vẫn tên tương tự như danh mục.
     1. Nút “sửa” tại các form danh sách điều chuyển là “Xem”
     2. Tạo mới: Click tạo mới tai form list hoặc Entry.
        1. Check quyền tạo mới của control “btnNew”. Nếu có quyền thị open form cho phép nhập thông tin tạo mới
        2. Cảnh báo nếu ko có quyền
        3. Nếu là Mode tạo mới thì nút “Lưu” gọi chức năng Insert không cần check quyền nút lưu.
     3. Sửa: khác với form danh muc “Sửa” form Entry của chứng từ là 1 mode riêng nên khi muốn sửa Click vào Button “Sửa” 🡺 check quyền button “ntnUpdate”.
        1. Nếu có quyền thì open form cho sửa thông tin
        2. Click vào lưu để lưu lại
        3. Nếu không thì cảnh báo không có quyền.
     4. Xóa thì check quyền control “btnDelete” để xác nhận quyền
     5. Các chức năng khác check quyền trước khi thực thi (In, Xuất, Nhập dữ liệu…)
  2. **Form danh mục thêm mới, cập nhật trên lưới.**
     1. **Thêm mới:** Check quyền button btnNew
     2. **Sửa:** Khi sửa trên lưới và lưu từng dòng hoặc nhấn vào nút lưu. Trước khi lưu thì check lại quyền Sửa control btnUpdate. Nút sửa sẽ ẩn đi vì quyền sửa sẽ được check trước khi lưu.

Nếu không có quyền thì thì cảnh báo và quay lại trang thái ban đầu của Record dang focus

* + 1. **Xóa.** Check quyền btnDelete