Software: Asset Declaration

Users: BPCL Staff

Reviewer: Rajesh Ambakatt

Developer: Harshavardhan Achrekar

Project Manager: S.K. Dwivedi

CMM Documents not attached
ASSET DECLARATION MENU

PLEASE READ THE INSTRUCTION BEFORE FILLING THE FORMS.

1. Submission of Annual Property / Returns of Assets and Liabilities
   For Submission of Assets and Liabilities of each type into our system.

2. View the Annual Property return
   For Viewing the Assets and Liabilities of each type as entered by Staff.

3. View My Declarations Status
   For Viewing the status of numbers of records of each type of assets and liabilities entered by Staff.

4. Close a Declaration Year
   For Closing the Current Declaration Year and Opening the next declaration Year.

INSTRUCTIONS TO BE FOLLOWED WHILE FILLING THE ASSETS AND LIABILITIES ARE LISTED BELOW:

1. Most of the text fields are having a length of 100 characters, so fill in short details.
2. On opening the asset declaration, you will find there are 6 forms to be filled.

   Form No. 1 - Immovable Properties
   Form No. 2 - A - Liquid Assets and Investments
   Form No. 2 - B - Non-liquid Assets
   Form No. 3 - Shares/Debentures/Issues
   Form No. 3 - Annuity (transactions > Rs 25000)

3. Before filling the form, read instructions carefully by clicking Instructions link on each form.
4. Fill the form correctly in all respect and save. If the entry does not appear in "View my declaration" i.e., lower grid, then look...
**FORM NO.2 A/B [b]No. of Case/Year[/b]**

**STAFF DETAILS:**

<table>
<thead>
<tr>
<th>Description</th>
<th>Amount / Original Price</th>
<th>Held in the name of &amp; relationship</th>
<th>How acquired</th>
<th>Annual income derived</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>X</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>X</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>X</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
</tbody>
</table>

**VIEW MY DECLARATION (FILE INVARIABLE ASSETS):**

<table>
<thead>
<tr>
<th>Edit</th>
<th>Sl. No</th>
<th>Description</th>
<th>Amount / Original Price</th>
<th>Held in the name of &amp; relationship</th>
<th>How acquired</th>
<th>Annual Income Derived</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
</tbody>
</table>

Reason for Deletion: 

**SAVE**
**FORM NO. 2C: |

**Staff Details:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Designation</th>
<th>Date of Joining</th>
<th>Basic Pay</th>
</tr>
</thead>
</table>

- **Liabilities to the Bank/Financial Institution**
- Liabilities to friends and relatives
- Other liabilities, if any

**Liabilities Details:**

<table>
<thead>
<tr>
<th>SL.NO.</th>
<th>Liabilities to the Bank/Financial Institution</th>
<th>Liabilities to friends and relatives</th>
<th>Other liabilities, if any</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Diners Club</td>
<td>Diners Club</td>
<td>Diners Club</td>
</tr>
<tr>
<td></td>
<td>Diners Club</td>
<td>Diners Club</td>
<td>Diners Club</td>
</tr>
<tr>
<td></td>
<td>Diners Club</td>
<td>Diners Club</td>
<td>Diners Club</td>
</tr>
<tr>
<td></td>
<td>Diners Club</td>
<td>Diners Club</td>
<td>Diners Club</td>
</tr>
</tbody>
</table>

**Reason for Deletion:** [Save]
<table>
<thead>
<tr>
<th>SL.Nos.</th>
<th>Details/Description of property and its location (see ReadMe in notes 1 &amp; 5) to whom it was held and his/her relationship, if any, to the employee</th>
<th>How and when acquired (See ReadMe in notes 2 &amp; 6)</th>
<th>Value of the Property (See ReadMe in notes 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>Sl.No</td>
<td>Description</td>
<td>Amount/Original Price</td>
<td>Held in the Name of &amp; Relationship</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-----------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>2</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>3</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>4</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>5</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>6</td>
<td>Dashboard</td>
<td>Dashboard</td>
<td>Dashboard</td>
</tr>
<tr>
<td>Name of company</td>
<td>No. of shares</td>
<td>Face value</td>
<td>Cost of acquisition</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Debtors</td>
<td>Debtors</td>
<td>Debtors</td>
<td>Debtors</td>
</tr>
<tr>
<td>Debtors</td>
<td>Debtors</td>
<td>Debtors</td>
<td>Debtors</td>
</tr>
<tr>
<td>Debtors</td>
<td>Debtors</td>
<td>Debtors</td>
<td>Debtors</td>
</tr>
</tbody>
</table>
CLOSE DECLARATION YEAR

Instructions to close a Declaration Year

Note 1. Once the declaration year is closed you cannot enter assets for that particular year nor can you edit or delete them.

Note 2. After closing the declaration year, the records up to the closing year are copied into the next declaration year.

Note 3. Management Staff has the option to modify those copied records in the new declaration year but deletion of those records requires reason to be filled up by the staff.

Note 4. This Module serves as the exit point of the system.
Namespace AssetBusinessLogic

Public Class asset_immovable
    Private _Sno As Integer
    Private _staffno As Integer
    Private _decl_year As String
    Private _prop_desc_loc As String
    Private _owner_name_rel As String
    Private _acquire_details As String
    Private _prop_value As String
    Private _annual_income_derived As String
    Private _remark As String
    Private _decl_status As Char
    Private _update_date As Date
    Private _enter_by As String
    Private _enter_date As Date
    'Private _DelRemark As String

    Public Property Sno() As Integer
        Get
            Return _Sno
        End Get
        Set(ByVal Value As Integer)
            _Sno = Value
        End Set
    End Property
End Class
Public Property staffno() As Integer
    Get
        Return _staffno
    End Get
    Set(ByVal Value As Integer)
        _staffno = Value
    End Set
End Property
Public Property decl_year() As String
    Get
        Return _decl_year
    End Get
    Set(ByVal Value As String)
        _decl_year = Value
    End Set
End Property
Public Property prop_desc_loc() As String
    Get
        Return _prop_desc_loc
    End Get
    Set(ByVal Value As String)
        _prop_desc_loc = Value
    End Set
End Property
Public Property owner_name_rel() As String
    Get
        Return _owner_name_rel
    End Get
    Set(ByVal Value As String)
        _owner_name_rel = Value
    End Set
End Property
Public Property acquire_details() As String
    Get
        Return _acquire_details
    End Get
    Set(ByVal Value As String)
        _acquire_details = Value
    End Set
End Property
Public Property prop_value() As String
    Get
        Return _prop_value
    End Get
    Set(ByVal Value As String)
        _prop_value = Value
    End Set
End Property
Public Property annual_income Derived() As String
    Get
        Return _annual_income_derived
    End Get
    Set(ByVal Value As String)
        _annual_income_derived = Value
    End Set
End Property
Public Property remark() As String
Get
    Return _remark
End Get
Set(ByVal Value As String)
    _remark = Value
End Set
End Property
Public Property decl_status() As Char
Get
    Return _decl_status
End Get
Set(ByVal Value As Char)
    _decl_status = Value
End Set
End Property
Public Property Update_date() As Date
Get
    Return _Update_date
End Get
Set(ByVal Value As Date)
    _Update_date = Value
End Set
End Property
Public Property enter_by() As String
Get
    Return _enter_by
End Get
Set(ByVal Value As String)
    _enter_by = Value
End Set
End Property
Public Property enter_date() As Date
Get
    Return _enter_date
End Get
Set(ByVal Value As Date)
    _enter_date = Value
End Set
End Property
'Public Property DelRemark() As String
'    Get
'        Return _DelRemark
'    End Get
'    Set(ByVal Value As String)
'        _DelRemark = Value
'    End Set
'End Property
Public Sub New()
End Sub
Public Sub New(ByVal TStaffno As Integer)
    _staffno = TStaffno
End Sub
Public Sub New(ByVal Tstaffno As Integer, ByVal Tdecl_year As String, ByVal Tprop_desc_loc As String, ByVal Towner_name_rel As String, ByVal Tacquire_details As String, ByVal Tprop_value As String, ByVal Tannual_income Derived As String, ByVal Tremark As String, ByVal Tremark As String)
Public Function GetMyimmovableAssetsDetails() As asset_immovablecollections
    Dim dsimmovableasset As DataSet = SqlHelper.ExecuteDataset(ConnectionString, "Usp_AR_GetMyimmovableAssetsDetails", _staffno, _decl_year)
    Dim drow As DataRow
    Dim Timmovableassetcoll As New asset_immovablecollections
    For Each drow In dsimmovableasset.Tables(0).Rows
        Dim Newimmovableasset As New asset_immovable
        Newimmovableasset.Sno = Integer.Parse(drow.Item("Sno").ToString())
        Newimmovableasset.staffno = Integer.Parse(drow.Item("staffno").ToString())
        Newimmovableasset.decl_year = drow.Item("decl_year").ToString()
        Newimmovableasset.prop_desc_loc = drow.Item("prop_desc_loc").ToString()
        Newimmovableasset.owner_name_rel = drow.Item("owner_name_rel").ToString()
        Newimmovableasset.acquire_details = drow.Item("acquire_details").ToString()
        Newimmovableasset.prop_value = drow.Item("prop_value").ToString()
        Newimmovableasset.annual_income_derived = drow.Item("annual_income_derived").ToString()
        Newimmovableasset.remark = drow.Item("remark").ToString()
        Newimmovableasset.decl_status = Char.Parse(drow.Item("decl_status").ToString())
        Newimmovableasset.Update_date = Date.Parse(drow.Item("Update_date").ToString())
        Newimmovableasset.enter_by = drow.Item("enter_by").ToString()
        Newimmovableasset.enter_date = Date.Parse(drow.Item("enter_date").ToString())
        ' Newimmovableasset.DelRemark = drow.Item("DelRemark")
        Timmovableassetcoll.Add(Newimmovableasset)
    Next
    Return Timmovableassetcoll
End Function

Public Function addMyimmovableAssetsDetails() As Integer
Dim Result As Integer
Result = Convert.ToInt32(SqlHelper.ExecuteScalar(ConnectionString, "Usp_AR_addMyimmovableAssetsDetails", _staffno, _decl_year, _prop_desc_loc, _owner_name_rel, _acquire_details, _prop_value, _annual_income Derived, _remark, _decl_status, _Update_date, _enter_by, _enter_date), DelRemark))
    addMyimmovableAssetsDetails = Result
End Function
Public Function updateMyimmovableAssetsDetails() As Integer
    Dim Result As Integer
    Result = Convert.ToInt32(SqlHelper.ExecuteScalar(ConnectionString, "Usp_AR_updateMyimmovableAssetsDetails", _Sno, _staffno, _decl_year, _prop_desc_loc, _owner_name_rel, _acquire_details, _prop_value, _annual_income Derived, _remark, _decl_status, _Update_date, _enter_by, _enter_date), _DelRemark))
    updateMyimmovableAssetsDetails = Result
End Function
'Public Function deleteMyimmovableAssetsDetails() As Integer
    Dim Result As Integer
    Result = Convert.ToInt32(SqlHelper.ExecuteScalar(ConnectionString, "Usp_AR_deleteMyimmovableAssetsDetails", _Sno))
    deleteMyimmovableAssetsDetails = Result
'End Function
End Class
End Namespace

Namespace AssetBusinessLogic
    Public Class asset_immovablecollections
        Inherits ArrayList
    End Class
End Namespace

Module Module1
Imports System.IO
Imports System.Xml
Imports System.Text
Namespace AssetBusinessLogic
    Module Module1
    'Public ConnectionString As String = "Persist Security
Info=False;User ID=webuser;Pwd=webuser;Initial
Catalog=CentralMedical;Data Source=chomddat1"
    Public ConnectionString As String = Getxmlstr() 'FROM XML FILE
    'Public ConnectionString As String = Gettextstr() 'FROM TEXT FILE
    Public Function Getxmlstr() As String
        Dim Reader As XmlTextReader
        Dim Mystr As String
        Reader = New XmlTextReader("D:\SqlConnection\ConnectionData.xml")
        While Reader.Read()
            Reader.MoveToContent()
    End Function
End Module
End Namespace
If (Reader.NodeType = System.Xml.XmlNodeType.Element And Reader.Name = "ConnectionString") Then
    Mystr = Reader.ReadString()
End If
End While
Dim StrBuild As New StringBuilder(Mystr)
StrBuild.Append(";Initial Catalog=HRS_Admin")
Getxmlstr = StrBuild.ToString
End Function
Public Function Gettextstr() As String
    Dim Filestreams As StreamReader
    Dim Mystr As String
    Filestreams = File.OpenText("D:\SqlConnection\ConnectionData.txt")
    Mystr = Filestreams.ReadLine()
    Dim StrBuild As New StringBuilder(Mystr)
    StrBuild.Append(";Initial Catalog=HRS_Admin")
    Gettexts = StrBuild.ToString
End Function
End Module
End Namespace

StaffDeclaration Class
Imports System.Web.Mail
Namespace AssetBusinessLogic

Public Class StaffDeclaration
    Private _Sno As Integer
    Private _staffno As Integer
    Private _decl_year As String
    Private _name As String
    Private _basic As Integer
    Private _count_immovable As Integer
    Private _count_movable_l As Integer
    Private _count_movable_m As Integer
    Private _count_liaabilities As Integer
    Private _count_shares As Integer
    Private _status As Integer
    Private _Dossier_owner As String
    Private _Dossier_status As Char
    Private _desig As String
    Private _doj_mgmt As Date
    Private _loc_cd As String
    Private _loc_nm As String

    Public Property staffno() As Integer
        Get
            Return _staffno
        End Get
        Set(ByVal Value As Integer)
            _staffno = Value
        End Set
    End Property

    Public Property decl_year() As String
        Get
            Return _decl_year
        End Get
    End Property

End Class
End Namespace
Set(ByVal Value As String)
    _decl_year = Value
End Set
End Property
Public Property name() As String
    Get
        Return _name
    End Get
    Set(ByVal Value As String)
        _name = Value
    End Set
End Property
Public Property basic() As Integer
    Get
        Return _basic
    End Get
    Set(ByVal Value As Integer)
        _basic = Value
    End Set
End Property
Public Property count_immovable() As Integer
    Get
        Return _count_immovable
    End Get
    Set(ByVal Value As Integer)
        _count_immovable = Value
    End Set
End Property
Public Property count_movable_l() As Integer
    Get
        Return _count_movable_l
    End Get
    Set(ByVal Value As Integer)
        _count_movable_l = Value
    End Set
End Property
Public Property count_movable_m() As Integer
    Get
        Return _count_movable_m
    End Get
    Set(ByVal Value As Integer)
        _count_movable_m = Value
    End Set
End Property
Public Property count_liabilities() As Integer
    Get
        Return _count_liabilities
    End Get
    Set(ByVal Value As Integer)
        _count_liabilities = Value
    End Set
End Property
Public Property count_shares() As Integer
    Get
        Return _count_shares
    End Get
    Set(ByVal Value As Integer)
Public Property count_shares() As Integer
Get
    Return _count_shares
End Get
Set(ByVal Value As Integer)
    _count_shares = Value
End Set
End Property

Public Property status() As Integer
Get
    Return _status
End Get
Set(ByVal Value As Integer)
    _status = Value
End Set
End Property

Public Property Dossier_owner() As String
Get
    Return _Dossier_owner
End Get
Set(ByVal Value As String)
    _Dossier_owner = Value
End Set
End Property

Public Property Dossier_status() As Char
Get
    Return _Dossier_status
End Get
Set(ByVal Value As Char)
    _Dossier_status = Value
End Set
End Property

Public Property desig() As String
Get
    Return _desig
End Get
Set(ByVal Value As String)
    _desig = Value
End Set
End Property

Public Property doj_mgmt() As Date
Get
    Return _doj_mgmt
End Get
Set(ByVal Value As Date)
    _doj_mgmt = Value
End Set
End Property

Public Property loc_cd() As String
Get
    Return _loc_cd
End Get
Set(ByVal Value As String)
    _loc_cd = Value
End Set
End Property

Public Property loc_nm() As String
Get
    Return _loc_nm
End Get
Set(ByVal Value As String)
    _loc_nm = Value
End Set
End Property
Public Sub New()
End Sub
Public Sub New(ByVal Tstaffno As Integer, ByVal Tdecl_year As String, ByVal Tname As String, ByVal Tbasic As Integer, ByVal Tcount_immovable As Integer, ByVal Tcount_movable_l As Integer, ByVal Tcount_movable_m As Integer, ByVal Tcount_liabilities As Integer, ByVal Tcount_shares As Integer, ByVal Tstatus As Integer, ByVal TDossier_owner As String, ByVal TDossier_status As Char, ByVal Tdesig As String, ByVal Tdoj_mgmt As Date, ByVal Tloc_cd As String, ByVal Tloc_nm As String)
    _staffno = Tstaffno
    _decl_year = Tdecl_year
    _name = Tname
    _basic = Tbasic
    _count_immovable = Tcount_immovable
    _count_movable_l = Tcount_movable_l
    _count_movable_m = Tcount_movable_m
    _count_liabilities = Tcount_liabilities
    _count_shares = Tcount_shares
    _status = Tstatus
    _Dossier_owner = TDossier_owner
    _Dossier_status = TDossier_status
    _desig = Tdesig
    _doj_mgmt = Tdoj_mgmt
    _loc_cd = Tloc_cd
    _loc_nm = Tloc_nm
End Sub
Public Function GetMyStaffDeclarationDetails(ByVal Tstaffno As Integer) As StaffDeclarationCollection
    Dim dsmovableasset As DataSet = SqlHelper.ExecuteDataset(ConnectionString, "Usp_AR_GetAllStaffAssetsDeclarationDetails")
    Dim drow As DataRow
    Dim Tmovableassetcoll As New StaffDeclarationCollection
    For Each drow In dsmovableasset.Tables(0).Rows
        Dim Newmovableasset As New StaffDeclaration
        Newmovableasset.staffno = Integer.Parse(drow.Item("staffno").ToString())
        Newmovableasset.decl_year = drow.Item("decl_year").ToString()
        Newmovableasset.name = drow.Item("name").ToString()
        Newmovableasset.basic = Integer.Parse(drow.Item("basic").ToString())
        Newmovableasset.count_immovable = Integer.Parse(drow.Item("count_immovable").ToString())
        Newmovableasset.count_movable_l = Integer.Parse(drow.Item("count_movable_l").ToString())
        Newmovableasset.count_movable_m = Integer.Parse(drow.Item("count_movable_m").ToString())
        Newmovableasset.count_liabilities = Integer.Parse(drow.Item("count_liabilities").ToString())
        Newmovableasset.count_shares = Integer.Parse(drow.Item("count_shares").ToString())
        Newmovableasset.status = Integer.Parse(drow.Item("status").ToString())
        Tmovableassetcoll.Add(Newmovableasset)
    Next
    Return Tmovableassetcoll
End Function
Newmovableasset.Dossier_owner = drow.Item("Dossier_owner").ToString()
Newmovableasset.Dossier_status = Char.Parse(drow.Item("Dossier_status").ToString())
Newmovableasset.desig = drow.Item("desig").ToString()
Newmovableasset.doj_mgmt = Date.Parse(drow.Item("doj_mgmt").ToString())
Newmovableasset.loc_cd = drow.Item("loc_cd").ToString()
Newmovableasset.loc_nm = drow.Item("loc_nm").ToString()
If Newmovableasset.staffno = Tstaffno Then
    Tmovableassetcoll.Add(Newmovableasset)
End If
Next
Return Tmovableassetcoll
End Function
Public Function addMyStaffDeclarationDetails() As Integer
    Dim Result As Integer
    Result = Convert.ToInt32(SqlHelper.ExecuteScalar(ConnectionString, "Usp_AR_addMyAssetsDeclarationDetails", _staffno, _decl_year, _name, _basic, _count_immovable, _count_movable_l, _count_movable_m, _count_liabilities, _count_shares, status, _Dossier_owner, _Dossier_status, _desig, _doj_mgmt, _loc_cd, _loc_nm))
    addMyStaffDeclarationDetails = Result
End Function
Public Function closeMyStaffDeclarationDetails() As Integer
    Dim Result As Integer
    Result = Convert.ToInt32(SqlHelper.ExecuteScalar(ConnectionString, "Usp_AR_closeMyAssetsDeclarationDetails", _staffno, _decl_year, _status))
    closeMyStaffDeclarationDetails = Result
End Function
Public Shared Sub SendMail(ByVal strFrom As String, ByVal strTo As String, ByVal strSubject As String, ByVal strBody As String)
    Dim objx As HttpContext = HttpContext.Current
    Dim objmessage As New MailMessage
    objmessage.From = strFrom
    objmessage.To = strTo
    objmessage.Subject = strSubject
    objmessage.Body = strBody
    Dim strExchangeServer As String = CType(objx.Application("Exchangeserver"), String)
    'Use HTML formatted e-mail
    objmessage.BodyFormat = MailFormat.Html
    SmtpMail.SmtpServer = strExchangeServer
    "cochomfes1.corp.bharatpetroleum.com" 'ToDo hard code
    SmtpMail.Send(objmessage)
End Sub
End Class
End Namespace

Namespace AssetBusinessLogic
    Public Class StaffDeclarationCollection
        Inherits ArrayList
    End Class
End Namespace
Sql Helper Class Code

'*********************************************************************
' Microsoft Data Access Application Block for .NET
'
'SQLHelper.cs
'
'This file contains the implementations of the SqlHelper and
SqlHelperParameterCache
'classes.
'
'For more information see the Data Access Application Block
Implementation Overview.
'
'*********************************************************************

Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Collections

'*********************************************************************

Private Sub New()  
End Sub 'New

'*********************************************************************

' The SqlHelper class is intended to encapsulate high performance,
scalable best practices for
'common uses of SqlClient.
'
'*********************************************************************

Public NotInheritable Class SqlHelper

'*********************************************************************

' Since this class provides only static methods, make the default
constructor private to prevent
'instances from being created with "new SqlHelper()".
'
'*********************************************************************
This method will assign a value of DBNull to any parameter with a direction of InputOutput and a value of null.

This behavior will prevent default values from being used, but this will be the less common case than an intended pure output parameter (derived as InputOutput) where the user provided no input value.

param name="command" The command to which the parameters will be added
param name="commandParameters" an array of SqlParameters tho be added to command

*********************************************************************
Private Shared Sub AttachParameters(ByVal command As SqlCommand, ByVal commandParameters() As SqlParameter)  Dim p As SqlParameter  For Each p In commandParameters  'check for derived output value with no value assigned  If p.Direction = ParameterDirection.InputOutput And p.Value Is Nothing Then  p.Value = Nothing  End If  command.Parameters.Add(p)  Next p  End Sub 'AttachParameters

*********************************************************************
This method assigns an array of values to an array of SqlParameters.

param name="commandParameters" array of SqlParameters to be assigned values
param name="parameterValues" array of objects holding the values to be assigned

*********************************************************************
Private Shared Sub AssignParameterValues(ByVal commandParameters() As SqlParameter, ByVal parameterValues() As Object)  Dim i As Short  Dim j As Short

If (commandParameters Is Nothing) And (parameterValues Is Nothing) Then  'do nothing if we get no data  Return  End If

' we must have the same number of values as we pave parameters to put them in
If commandParameters.Length <> parameterValues.Length Then
    Throw New ArgumentException("Parameter count does not match Parameter Value count.")
End If

' value array
j = Convert.ToInt16(commandParameters.Length - 1)
For i = 0 To j
    commandParameters(i).Value = parameterValues(i)
Next

End Sub 'AssignParameterValues

*********************************************************************
' This method opens (if necessary) and assigns a connection, transaction, command type and parameters to the provided command.
' param name="command" the SqlCommand to be prepared
' param name="connection" a valid SqlConnection, on which to execute this command
' param name="transaction" a valid SqlTransaction, or 'null'
' param name="commandType" the CommandType (stored procedure, text, etc.)
' param name="commandText" the stored procedure name or T-SQL command
' param name="commandParameters" an array of SqlParameters to be associated with the command or 'null' if no parameters are required
*********************************************************************

Private Shared Sub PrepareCommand(ByVal command As SqlCommand, ByVal connection As SqlConnection, ByVal transaction As SqlTransaction, ByVal commandType As CommandType, ByVal commandText As String, ByVal commandParameters() As SqlParameter)

    ' if the provided connection is not open, we will open it
If connection.State <> ConnectionState.Open Then
    connection.Open()
End If

    ' associate the connection with the command
    command.Connection = connection

    ' set the command text (stored procedure name or SQL statement)
    command.CommandText = commandText

    ' if we were provided a transaction, assign it.
    If Not (transaction Is Nothing) Then

command.Transaction = transaction
End If

'set the command type
command.CommandType = commandType

'attach the command parameters if they are provided
If Not (commandParameters Is Nothing) Then
    AttachParameters(command, commandParameters)
End If

Return
End Sub 'PrepareCommand

'***********************************************************************************
' Execute a SqlCommand (that returns no resultset) against the
database specified in the connection string
' using the provided parameters.
' e.g.:
' int result = ExecuteNonQuery(connString,
CommandType.StoredProcedure, "PublishOrders", new
SqlParameter("@prodid", 24));
' param name="connectionString" a valid connection string for a
SqlConnection
' param name="commandType" the CommandType (stored procedure, text,
etc.)
' param name="commandText" the stored procedure name or T-SQL command
' param name="commandParameters" an array of SqlParameters used to
execute the command
' returns an int representing the number of rows affected by the
command
'
'***********************************************************************************

Public Overloads Shared Function ExecuteNonQuery(ByVal connectionString As String, _
    ByVal commandType As CommandType, _
    ByVal commandText As String, _
    ByVal commandParameters() As SqlParameter) As Integer
    'create & open a SqlConnection, and dispose of it after we are
done.
    Dim cn As New SqlConnection(connectionString)
    Try
        cn.Open()
        'call the overload that takes a connection in place of the
connection string
Return ExecuteNonQuery(cn, commandType, commandText, commandParameters)
Finally
    cn.Dispose()
End Try
End Function 'ExecuteNonQuery

Public Overloads Shared Function ExecuteNonQuery(ByVal connectionString As String, ByVal spName As String, ByVal parameterValues() As Object) As Integer
    Dim commandParameters As SqlParameter()
    'if we receive parameter values, we need to figure out where they go
    If Not (parameterValues Is Nothing) And parameterValues.Length > 0 Then
        'pull the parameters for this stored procedure from the parameter cache (or discover them & populate the cache)
        commandParameters = SqlHelperParameterCache.GetSpParameterSet(connectionString, spName)
        'assign the provided values to these parameters based on parameter order
        AssignParameterValues(commandParameters, parameterValues)
'call the overload that takes an array of SqlParameter
Return ExecuteNonQuery(connectionString,
CommandType.StoredProcedure, spName, commandParameters)
'otherwise we can just call the SP without params
Else
Return ExecuteNonQuery(connectionString,
CommandType.StoredProcedure, spName)
End If
End Function 'ExecuteNonQuery

*********************************************************************
'* Execute a SqlCommand (that returns no resultset) against the
specified SqlConnection
' using the provided parameters.
'* e.g.:
'int result = ExecuteNonQuery(conn, CommandType.StoredProcedure,
"PublishOrders", new SqlParameter("@prodid", 24));
'* param name="connection" a valid SqlConnection
' param name="commandType" the CommandType (stored procedure, text, etc.)
' param name="commandText" the stored procedure name or T-SQL command
' param name="commandParameters" an array of SqlParameter used to
execute the command
' returns an int representing the number of rows affected by the
command

*********************************************************************
Public Overloads Shared Function ExecuteNonQuery(ByVal connection
As SqlConnection, _
ByVal commandType
As CommandType, _
ByVal commandText
As String, _
ByVal commandParameters() As SqlParameter) As Integer

'create a command and prepare it for execution
Dim cmd As New SqlCommand
Dim retval As Integer

PrepareCommand(cmd, connection, CType(Nothing, SqlTransaction),
commandType, commandText, commandParameters)

'finally, execute the command.
retval = cmd.ExecuteNonQuery()

'detach the SqlParameter from the command object, so they can
be used again
cmd.Parameters.Clear()

Return retval
End Function 'ExecuteNonQuery

'*********************************************************************
' Execute a SqlCommand (that returns a resultset) against the
database specified in the connection string
' using the provided parameters.
' e.g.:
'  DataSet ds = ExecuteDataset(connString,
 CommandType.StoredProcedure, "GetOrders", new SqlParameter("@prodid",
 24));
'  param name="connectionString" a valid connection string for a
SqlConnection
'  param name="commandType" the CommandType (stored procedure, text,
  etc.)
'  param name="commandText" the stored procedure name or T-SQL
  command
'  param name="commandParameters" an array of SqlParameters used to
  execute the command
'  returns a dataset containing the resultset generated by the
command
'
'*********************************************************************

Public Overloads Shared Function ExecuteDataset(ByVal
connectionString As String, _
ByVal commandType
As CommandType, _
ByVal commandText
As String, _
ByVal commandParameters() As SqlParameter) As DataSet
  'create & open a SqlConnection, and dispose of it after we are
done.
  Dim cn As New SqlConnection(connectionString)
  Try
    cn.Open()
    'call the overload that takes a connection in place of the
   connection string
    Return ExecuteDataset(cn, commandType, commandText,
    commandParameters)
  Finally
    cn.Dispose()
  End Try
End Function 'ExecuteDataset

'*********************************************************************
'
' Execute a stored procedure via a SqlCommand (that returns a resultset) against the database specified in the connection string using the provided parameter values. This method will query the database to discover the parameters for the stored procedure (the first time each stored procedure is called), and assign the values based on parameter order.

This method provides no access to output parameters or the stored procedure's return value parameter.

e.g.:
保护区.main = ExecuteDataset(connString, "GetOrders", 24, 36);

param name="connectionString" a valid connection string for a SqlConnection
param name="spName" the name of the stored procedure
param name="parameterValues" an array of objects to be assigned as the input values of the stored procedure
returns a dataset containing the resultset generated by the command

********************************************************************************
Public Overloads Shared Function ExecuteDataset(ByVal connectionString As String, _
String, _
ByVal spName As _
parameterValues() As Object) As DataSet

    Dim commandParameters As SqlParameter()
    
    'if we receive parameter values, we need to figure out where they go
    If Not (parameterValues Is Nothing) And parameterValues.Length > 0 Then
        'pull the parameters for this stored procedure from the parameter cache (or discover them & populate the cache)
        commandParameters = SqlHelperParameterCache.GetSpParameterSet(connectionString, spName)
        
        'assign the provided values to these parameters based on parameter order
        AssignParameterValues(commandParameters, parameterValues)
        
        'call the overload that takes an array of SqlParameters
        Return ExecuteDataset(connectionString, CommandType.StoredProcedure, spName, commandParameters)
    Else
        'otherwise we can just call the SP without params
        Return ExecuteDataset(connectionString, CommandType.StoredProcedure, spName)
    End If

End Function 'ExecuteDataset

********************************************************************************
' Execute a SqlCommand (that returns a resultset) against the
specified SqlConnection
' using the provided parameters.
' e.g.:
'  DataSet ds = ExecuteDataset(conn, CommandType.StoredProcedure,
"GetOrders", new SqlParameter("@prodid", 24));
' param name="connection" a valid SqlConnection
' param name="commandType" the CommandType (stored procedure, text,
etc.)
' param name="commandText" the stored procedure name or T-SQL
command
' param name="commandParameters" an array of SqlParameters used to
execute the command
' returns a dataset containing the resultset generated by the
command

'*********************************************************************
Public Overloads Shared Function ExecuteDataset(ByVal connection As
SqlConnection, _
ByVal commandType As CommandType, _
ByVal commandText As String, _
ByVal ParamArray commandParameters() As SqlParameter) As DataSet

    'create a command and prepare it for execution
    Dim cmd As New SqlCommand
    Dim ds As New DataSet
    Dim da As SqlDataAdapter

    PrepareCommand(cmd, connection, CType(Nothing, SqlTransaction),
commandType, commandText, commandParameters)

    'create the DataAdapter & DataSet
    da = New SqlDataAdapter(cmd)

    'fill the DataSet using default values for DataTable names,
    etc.
    da.Fill(ds)

    'detach the SqlParameters from the command object, so they can
be used again
    cmd.Parameters.Clear()

    'return the dataset
    Return ds

End Function 'ExecuteDataset

'*********************************************************************
Execute a SqlCommand (that returns a 1x1 resultset) against the database specified in the connection string using the provided parameters.

  e.g.:
  int orderCount = (int)ExecuteScalar(connString, CommandType.StoredProcedure, "GetOrderCount", new SqlParameter("@prodid", 24));

  param name="connectionString" a valid connection string for a SqlConnection
  param name="commandType" the CommandType (stored procedure, text, etc.)
  param name="commandText" the stored procedure name or T-SQL command
  param name="commandParameters" an array of SqlParameters used to execute the command
  returns an object containing the value in the 1x1 resultset generated by the command

*********************************************************************
Public Overloads Shared Function ExecuteScalar(ByVal connectionString As String, _
  ByVal commandType As CommandType, _
  ByVal commandText As String, _
  ByVal ParamArray commandParameters() As SqlParameter) As Object
  'create & open a SqlConnection, and dispose of it after we are done.
  Dim cn As New SqlConnection(connectionString)
  Try
    cn.Open()

    'call the overload that takes a connection in place of the connection string
    Return ExecuteScalar(cn, commandType, commandText, commandParameters)
  Finally
    cn.Dispose()
  End Try
End Function 'ExecuteScalar

*********************************************************************

; ' Execute a stored procedure via a SqlCommand (that returns a 1x1 resultset) against the database specified in
; the connection string using the provided parameter values. This method will query the database to discover the parameters for the
; stored procedure (the first time each stored procedure is called), and assign the values based on parameter order.

This method provides no access to output parameters or the stored procedure's return value parameter.

e.g.:
```csharp
int orderCount = (int)ExecuteScalar(connString, "GetOrderCount", 24, 36);
```

- param name="connectionString" a valid connection string for a SqlConnection
- param name="spName" the name of the stored procedure
- param name="parameterValues" an array of objects to be assigned as the input values of the stored procedure
- returns an object containing the value in the 1x1 resultset generated by the command

*********************************************************************
Public Overloads Shared Function ExecuteScalar(ByVal connectionString As String, _
ByVal spName As String, _
ByVal parameterValues() As Object) As Object

Dim commandParameters As SqlParameter()

'if we receive parameter values, we need to figure out where they go
If Not (parameterValues Is Nothing) And parameterValues.Length > 0 Then
  'pull the parameters for this stored procedure from the parameter cache (or discover them & populate the cache)
  commandParameters = SqlHelperParameterCache.GetSpParameterSet(connectionString, spName)

  'assign the provided values to these parameters based on parameter order
  AssignParameterValues(commandParameters, parameterValues)

  'call the overload that takes an array of SqlParameters
  Return ExecuteScalar(connectionString, CommandType.StoredProcedure, spName, commandParameters)
Else
  'otherwise we can just call the SP without params
  Return ExecuteScalar(connectionString, CommandType.StoredProcedure, spName)
End If
End Function 'ExecuteScalar

*********************************************************************

Execute a SqlCommand (that returns a 1x1 resultset) against the specified SqlConnection
using the provided parameters.
' e.g.:
'    int orderCount = (int)ExecuteScalar(conn,
CommandType.StoredProcedure, "GetOrderCount", new
SqlParameter("@prodid", 24));
'    param name="connection" a valid SqlConnection
    param name="commandType" the CommandType (stored procedure, text, etc.)
    param name="commandText" the stored procedure name or T-SQL command
    param name="commandParameters" an array of SqlParameters used to execute the command
'    returns an object containing the value in the 1x1 resultset generated by the command

'********************************************************************************
Public Overloads Shared Function ExecuteScalar(ByVal connection As SqlConnection,
    ByVal commandType As CommandType, _
    ByVal commandText As String, _
    ByVal commandParameters() As SqlParameter) As Object
'create a command and prepare it for execution
    Dim cmd As New SqlCommand
    Dim retval As Object
    PrepareCommand(cmd, connection, CType(Nothing, SqlTransaction), commandType, commandText, commandParameters)
'execute the command & return the results
    retval = cmd.ExecuteScalar()
'detach the SqlParameters from the command object, so they can be used again
    cmd.Parameters.Clear()
    Return retval
End Function 'ExecuteScalar
End Class 'SqlHelper

'********************************************************************************

' SqlHelperParameterCache provides functions to leverage a static cache of procedure parameters, and the ability to discover parameters for stored procedures at run-time.
'********************************************************************************
Public NotInheritable Class SqlHelperParameterCache
Since this class provides only static methods, make the default constructor private to prevent instances from being created with "new SqlHelperParameterCache()".

*********************************************************************
Private Sub New()
End Sub 'New

Private Shared paramCache As Hashtable = Hashtable.Synchronized(New Hashtable)

*********************************************************************
' resolve at run time the appropriate set of SqlParameters for a stored procedure
' param name="connectionString" a valid connection string for a SqlConnection
' param name="spName" the name of the stored procedure
' param name="includeReturnValueParameter" whether or not to include their return value parameter

*********************************************************************
Private Shared Function DiscoverSpParameterSet(ByVal connectionString As String, ByVal spName As String, ByVal includeReturnValueParameter As Boolean, ByVal parameterValues() As Object) As SqlParameter()

Dim cn As New SqlConnection(connectionString)
Dim cmd As SqlCommand = New SqlCommand(spName, cn)
Dim discoveredParameters() As SqlParameter

Try
cn.Open()
cmd.CommandType = CommandType.StoredProcedure
SqlCommandBuilder.DeriveParameters(cmd)
If Not includeReturnValueParameter Then
cmd.Parameters.RemoveAt(0)
End If

discoveredParameters = New SqlParameter(cmd.Parameters.Count - 1) {}
cmd.Parameters.CopyTo(discoveredParameters, 0)
Finally
cmd.Dispose()
cn.Dispose()

End Try
Return discoveredParameters

End Function 'DiscoverSpParameterSet

'deep copy of cached SqlParameter array
Private Shared Function CloneParameters(ByVal originalParameters() As SqlParameter) As SqlParameter()

Dim i As Short
Dim j As Short = Convert.ToInt16(originalParameters.Length - 1)
Dim clonedParameters(j) As SqlParameter

For i = 0 To j
    clonedParameters(i) = CType(CType(originalParameters(i), ICloneable).Clone, SqlParameter)
Next

Return clonedParameters
End Function 'CloneParameters

'*********************************************************************
' add parameter array to the cache
' param name="connectionString" a valid connection string for a SqlConnection
' param name="commandText" the stored procedure name or T-SQL command
' param name="commandParameters" an array of SqlParameter to be cached
'

Public Shared Sub CacheParameterSet(ByVal connectionString As String, ByVal commandText As String, ByVal commandParameters() As SqlParameter)

Dim hashKey As String = connectionString +":"+ commandText
paramCache(hashKey) = commandParameters
End Sub 'CacheParameterSet

'*********************************************************************
' Retrieve a parameter array from the cache
' param name="connectionString" a valid connection string for a SqlConnection
' param name="commandText" the stored procedure name or T-SQL command
' returns an array of SqlParameter
'
Public Shared Function GetCachedParameterSet(ByVal connectionString As String, ByVal commandText As String) As SqlParameter()
    Dim hashKey As String = connectionString + ":" + commandText
    Dim cachedParameters As SqlParameter() = CType(paramCache(hashKey), SqlParameter())
    If cachedParameters Is Nothing Then
        Return Nothing
    Else
        Return CloneParameters(cachedParameters)
    End If
End Function 'GetCachedParameterSet

Public Overloads Shared Function GetSpParameterSet(ByVal connectionString As String, ByVal spName As String) As SqlParameter()
    Return GetSpParameterSet(connectionString, spName, False)
End Function 'GetSpParameterSet

Public Overloads Shared Function GetSpParameterSet(ByVal connectionString As String, ByVal spName As String, ByVal includeReturnValueParameter As Boolean) As SqlParameter()
    Return GetSpParameterSet(connectionString, spName, False)
End Function 'GetSpParameterSet
ByVal spName As String, _
ByVal includeReturnValueParameter As Boolean) As SqlParameter()

    Dim cachedParameters() As SqlParameter
    Dim hashKey As String

    hashKey = connectionString + ":" + spName + IIf(includeReturnValueParameter = True, ":include ReturnValue Parameter", ":").ToString

    cachedParameters = CType(paramCache(hashKey), SqlParameter())

    If (cachedParameters Is Nothing) Then
        paramCache(hashKey) = DiscoverSpParameterSet(connectionString, spName, includeReturnValueParameter)
        cachedParameters = CType(paramCache(hashKey), SqlParameter())
    End If

    Return CloneParameters(cachedParameters)

End Function 'GetSpParameterSet
End Class 'SqlHelperParameterCache

---

asset_open_decl_year.aspx

Imports assetdeclaration.AssetBusinessLogic
Public Class asset_open_decl_year
    Inherits System.Web.UI.Page
    Private ObjStaff As New Staff.StaffDetails
    Private Logon As String
    Private Staffno As Integer
    Private DsStaffInfo As DataSet

#Region " Web Form Designer Generated Code "

' This call is required by the Web Form Designer. 
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

    End Sub
    Protected WithEvents Label31 As System.Web.UI.WebControls.Label
    Protected WithEvents Hyperlink5 As System.Web.UI.WebControls.HyperLink
    Protected WithEvents Label129 As System.Web.UI.WebControls.Label
    Protected WithEvents Label130 As System.Web.UI.WebControls.Label
    Protected WithEvents Label127 As System.Web.UI.WebControls.Label
    Protected WithEvents Label128 As System.Web.UI.WebControls.Label
    Protected WithEvents Label125 As System.Web.UI.WebControls.Label
    Protected WithEvents Label123 As System.Web.UI.WebControls.Label
Protected WithEvents lbl_basicpay As System.Web.UI.WebControls.Label
Protected WithEvents Label21 As System.Web.UI.WebControls.Label
Protected WithEvents ddlCaseYear As System.Web.UI.WebControls.DropDownList
Protected WithEvents Button4 As System.Web.UI.WebControls.Button
Protected WithEvents lbl_result As System.Web.UI.WebControls.Label
Protected WithEvents Label3 As System.Web.UI.WebControls.Label
Protected WithEvents Label1 As System.Web.UI.WebControls.Label
Protected WithEvents Button1 As System.Web.UI.WebControls.Button

'NOTE: The following placeholder declaration is required by the Web Form Designer.
'Do not delete or move it.
Private designerPlaceholderDeclaration As System.Object

Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Init
'CODEGEN: This method call is required by the Web Form Designer
'Do not modify it using the code editor.
InitializeComponent()
End Sub

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
'Put user code to initialize the page here
If Page.IsPostBack = False Then
' Response.Redirect("ApplicationClose.htm")
' Response.End()
    Loadmyassest_Details()
End If
End Sub

'Private Sub get_yearlist()
'    Dim NewItem As New ListItem
'    NewItem.Value = Now.Year - 1
'    NewItem.Text = Now.Year - 1
'    ddlCaseYear.Items.Add(NewItem)
'    Dim NewItem1 As New ListItem
'    NewItem1.Value = 0
'    NewItem1.Text = "First"
'    ddlCaseYear.Items.Add(NewItem1)
'    'Dim fyearlist As New ArrayList
'    'fyearlist.Add(Now.Year)
'    'fyearlist.Add(Now.Year - 1)
'    'ddlCaseYear.DataSource = fyearlist
'    'ddlCaseYear.DataBind()
'    'ddlCaseYear.SelectedValue = (Now.Year - 1)
'End Sub

Private Sub Loadmyassest_Details()
    Logon = Request.ServerVariables("Logon_user")
    Staffno = ObjStaff.GetStaffNumber(Logon)
Dim Result As Integer
If Result = 2 Then
    Dim NewItem1 As New ListItem
    NewItem1.Value = "0"
    NewItem1.Text = "First"
    ddlCaseYear.Items.Add(NewItem1)
ElseIf Result = 1 Then
    Dim NewItem As New ListItem
    NewItem.Value = (Now.Year - 1).ToString
    NewItem.Text = (Now.Year - 1).ToString
    ddlCaseYear.Items.Add(NewItem)
Else
    Response.Redirect("ASSETMAIN.ASPX")
End If

DsStaffInfo = ObjStaff.GetBasicData(Staffno)
lbl_name.Text = DsStaffInfo.Tables(0).Rows(0).Item("Staff_nm").ToString
lbl_desig.Text = DsStaffInfo.Tables(0).Rows(0).Item("Desig").ToString
lbl_doj.Text = Format(DsStaffInfo.Tables(0).Rows(0).Item("Doj"), "dd/MMM/yyyy")
Dim dsData1 As DataSet = ObjStaff.GetPaymentDetails(Staffno)
lbl_basicpay.Text = CStr(dsData1.Tables(0).Rows(0).Item("amount"))

Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    Logon = Request.ServerVariables("Logon_user")
    Staffno = ObjStaff.GetStaffNumber(Logon)
    DsStaffInfo = ObjStaff.GetBasicData(Staffno)
Dim dossier_owner As String = DsStaffInfo.Tables(0).Rows(0).Item("dossier_owner").ToString()

Dim desig As String =
DsStaffInfo.Tables(0).Rows(0).Item("desig").ToString()
Dim doj_mgmt As Date = Date.Parse(DsStaffInfo.Tables(0).Rows(0).Item("doj_mgmt").ToString())
Dim loc_cd As String =
DsStaffInfo.Tables(0).Rows(0).Item("loc_cd").ToString()
Dim loc_nm As String =
DsStaffInfo.Tables(0).Rows(0).Item("loc_nm").ToString()

Dim decl_status As Integer
decl_status = 1
Dim Update_date As Date = Now.Date
Dim enter_date As Date = Now.Date

Dim C As Char = "1"
Dim StaffEntry As New StaffDeclaration(Staffno, ddlCaseYear.SelectedItem.Value, lbl_name.Text, CInt(lbl_basicpay.Text),
Dim RESULT As Integer = StaffEntry.addMyStaffDeclarationDetails() If RESULT = 1 Then
    lbl_result.Text = "You Have Successfully Opened the Declaration Year " & ddlCaseYear.SelectedItem.Text
    Response.Redirect("assetmain.aspx")
Else
    lbl_result.Text = "You cannot Open the Declaration Year " & ddlCaseYear.SelectedItem.Text
End If
End Sub
End Class

Imports assetdeclaration.AssetBusinessLogic
Public Class asset_immovable1
    Inherits System.Web.UI.Page

#Region " Web Form Designer Generated Code "
' This call is required by the Web Form Designer.
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
End Sub

Protected WithEvents Label18 As System.Web.UI.WebControls.Label
Protected WithEvents Label19 As System.Web.UI.WebControls.Label
 Protected WithEvents Label110 As System.Web.UI.WebControls.Label
Protected WithEvents Label12 As System.Web.UI.WebControls.Label
 Protected WithEvents lbl_name As System.Web.UI.WebControls.Label
Protected WithEvents Label11 As System.Web.UI.WebControls.Label
Protected WithEvents Label14 As System.Web.UI.WebControls.Label
Protected WithEvents Label13 As System.Web.UI.WebControls.Label
 Protected WithEvents dgvinput As System.Web.UI.WebControls.DataGrid
 Protected WithEvents Label15 As System.Web.UI.WebControls.Label
 Protected WithEvents HyperLink3 As System.Web.UI.WebControls.HyperLink
 Protected WithEvents HyperLink2 As System.Web.UI.WebControls.HyperLink
 Protected WithEvents lbl_ddlCaseYear As System.Web.UI.WebControls.Label
Protected WithEvents Label1 As System.Web.UI.WebControls.Label
 Protected WithEvents Button2 As System.Web.UI.WebControls.Button
 Protected WithEvents Label123 As System.Web.UI.WebControls.Label
 Protected WithEvents del_remarks As System.Web.UI.WebControls.TextBox
 Protected WithEvents Button1 As System.Web.UI.WebControls.Button
 Protected WithEvents RequiredFieldValidator1 As System.Web.UI.WebControls.RequiredFieldValidator
 Protected WithEvents Panel1 As System.Web.UI.WebControls.Panel
Protected WithEvents dtgriddisp As System.Web.UI.WebControls.DataGrid
Protected WithEvents Button3 As System.Web.UI.WebControls.Button
Protected WithEvents TextBox7 As System.Web.UI.WebControls.TextBox
Protected WithEvents Label24 As System.Web.UI.WebControls.Label
Protected WithEvents Panel2 As System.Web.UI.WebControls.Panel

'NOTE: The following placeholder declaration is required by the Web Form Designer.
'Do not delete or move it.
Private designerPlaceholderDeclaration As System.Object

Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Init
    'CODEGEN: This method call is required by the Web Form Designer
    'Do not modify it using the code editor.
    InitializeComponent()
    lbl_ddlCaseYear.Text = Request.Params("ddlCaseYear")
End Sub

#Region
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
    If Not Page.IsPostBack Then
        Panel2.Visible = False
        Loadassest_immovable()
        Loadmyassest_immovable()
    End If
End Sub
Private Sub Loadassest_immovable()
    Dim ObjStaff As New Staff.StaffDetails
    Dim Logon As String = Request.ServerVariables("Logon_user")
    Dim Staffno As Integer = ObjStaff.GetStaffNumber(Logon)
    Dim DsStaffInfo As DataSet = ObjStaff.GetB
    Dim Maxyear As Integer = Integer.Parse(SqlHelper.ExecuteScalar(ConnectionString, "Usp_AR_ChkMyAssetsDetails", Staffno).ToString())
    If (Maxyear > CInt(Request.Params("ddlCaseYear"))) Then
        Response.Write("You Cannot Enter this Details now")
        Response.End()
    End If
    If (Maxyear < CInt(Request.Params("ddlCaseYear"))) Then
        Response.Write("Please copy the Previous years Details into current year Before Proceeding to enter details of current year.Doing so your Previous year will be blocked for entry and updation.")
        Response.End()
        lbl_show.Text = "This is to notify you that u are entering details for this year for the first time after entering details for this year u cannot modify previous years records."
    Else
        If Convert.ToInt32(Session("DECLYEAR")) = 0 Then
            lbl_ddlCaseYear.Text = "FIRST"
        End If
    End If
lbl_show.Text = "You are Entering Immovable Assets for the year " + lbl_ddlCaseYear.Text
End If
Dim myasset_immovable As New asset_immovable(Staffno)
myasset_immovable.decl_year = Session("DECLYEAR").ToString()
dtgridinput.DataSource = myasset_immovable.GetMyimmovableAssetsDetails
dtgridinput.DataBind()
dtgridinput.Width = dtgridisp.Width
End Sub
Private Sub Loadmyassest_immovable()
Dim ObjStaff As New Staff.StaffDetails
Dim Logon As String = Request.ServerVariables("Logon_user")
Dim Staffno As Integer = ObjStaff.GetStaffNumber(Logon)
Dim DsStaffInfo As DataSet = ObjStaff.GetBasicData(Staffno)
lbl_name.Text = DsStaffInfo.Tables(0).Rows(0).Item("Staff_nm").ToString
lbl_desig.Text = DsStaffInfo.Tables(0).Rows(0).Item("Desig").ToString
lbl_doj.Text = Format(DsStaffInfo.Tables(0).Rows(0).Item("Doj"), "dd/MMM/yyyy")
Dim dsData1 As DataSet = ObjStaff.GetPaymentDetails(Staffno)
lbl_basicpay.Text = CStr(dsData1.Tables(0).Rows(0).Item("amount"))
Dim mydispassestimmovable As New asset_immovable(Staffno)
If lbl_ddlCaseYear.Text <> "FIRST" Then
    Session("DECLYEAR") = lbl_ddlCaseYear.Text()
Else
    Session("DECLYEAR") = 0
End If
mydispassestimmovable.decl_year = Session("DECLYEAR").ToString()
dtgriddisp.DataSource = mydispassestimmovable.GetMyimmovableAssetsDetails
dtgriddisp.DataBind()
dtgriddisp.Columns(1).Visible = False
End Sub
Private Sub dtgridinput_ItemCommand(ByVal source As System.Object, ByVal e As System.Web.UI.WebControls.DataGridCommandEventArgs) Handles dtgridinput.ItemCommand
If e.CommandName = "Insert" Then
    Dim decl_year As String
    If lbl_ddlCaseYear.Text <> "FIRST" Then
        Session("DECLYEAR") = lbl_ddlCaseYear.Text()
    Else
        Session("DECLYEAR") = 0
    End If
    decl_year = Session("DECLYEAR").ToString()
    Dim prop_desc_loc As String
    prop_desc_loc = CType(e.Item.FindControl("TextBox1"), TextBox).Text
    Dim owner_name_rel As String
    owner_name_rel = CType(e.Item.FindControl("TextBox2"), TextBox).Text
    Dim acquire_details As String
acquire_details = CType(e.Item.FindControl("TextBox3"), TextBox).Text

Dim prop_value As String = "0"
If CType(e.Item.FindControl("TextBox4"), TextBox).Text <> "" Then
    prop_value = CType(e.Item.FindControl("TextBox4"), TextBox).Text
End If

Dim annual_income_derived As String = "0"
If CType(e.Item.FindControl("TextBox5"), TextBox).Text <> "" Then
    annual_income_derived = CType(e.Item.FindControl("TextBox5"), TextBox).Text()
End If

Dim remark As String = ""
If CType(e.Item.FindControl("TextBox6"), TextBox).Text <> "" Then
    remark = CType(e.Item.FindControl("TextBox6"), TextBox).Text()
End If

Dim ObjStaff As New Staff.StaffDetails
Dim Logon As String = Request.ServerVariables("Logon_user")
Dim Staffno As Integer = ObjStaff.GetStaffNumber(Logon)

Dim DsStaffInfo As DataSet = ObjStaff.GetBasicData(Staffno)

Dim decl_status As Char
decl_status = "1"c
Dim Update_date As Date = Now.Date
Dim enter_date As Date = Now.Date
Dim myassetimmo As New asset_immovable(Staffno, decl_year, prop_desc_loc, owner_name_rel, acquire_details, prop_value, annual_income_derived, remark, decl_status, Update_date, Logon, enter_date)

myassetimmo.decl_year = Session("DECLYEAR").ToString
myassetimmo.addMyimmovableAssetsDetails()
dtgriddisp.EditItemIndex = -1
Loadmyasset_immovable()

CType(e.Item.FindControl("TextBox1"), TextBox).Text = ""
CType(e.Item.FindControl("TextBox2"), TextBox).Text = ""
CType(e.Item.FindControl("TextBox3"), TextBox).Text = ""
CType(e.Item.FindControl("TextBox4"), TextBox).Text = ""
CType(e.Item.FindControl("TextBox5"), TextBox).Text = ""
CType(e.Item.FindControl("TextBox6"), TextBox).Text = ""

End If
End Sub

    dtgriddisp.EditItemIndex = e.Item.ItemIndex
    Loadmyasset_immovable()
End Sub

dtgriddisp.EditItemIndex = -1
Loadmyasset_immovable()

    Dim strkey As String = CType(e.Item.FindControl("Label22"), Label).Text.ToString
    Dim decl_year As String
    If lbl_ddlCaseYear.Text <> "FIRST" Then
        Session("DECLYEAR") = lbl_ddlCaseYear.Text()
    Else
        Session("DECLYEAR") = 0
    End If
    decl_year = Session("DECLYEAR").ToString()
    Dim prop_desc_loc As String = CType(e.Item.FindControl("TextBox1"), TextBox).Text
    Dim owner_name_rel As String = CType(e.Item.FindControl("TextBox2"), TextBox).Text
    Dim acquire_details As String = CType(e.Item.FindControl("TextBox3"), TextBox).Text
    Dim prop_value As String = CType(e.Item.FindControl("TextBox4"), TextBox).Text
    Dim annual_income_derived As String = CType(e.Item.FindControl("TextBox5"), TextBox).Text
    Dim remark As String = CType(e.Item.FindControl("TextBox8"), TextBox).Text
    Dim ObjStaff As New Staff.StaffDetails
    Dim Logon As String = Request.ServerVariables("Logon_user")
    Dim Staffno As Integer = ObjStaff.GetStaffNumber(Logon)
    Dim dsStaffInfo As DataSet = ObjStaff.GetBasicData(Staffno)
    Dim decl_status As Char
    decl_status = "1"
    Dim Update_date As Date
    Update_date = Now.Date
    Dim enter_date As Date = Now.Date
    Dim myassetimmo As New asset_immovable(Staffno, decl_year, prop_desc_loc, owner_name_rel, acquire_details, prop_value, annual_income_derived, remark, decl_status, Update_date, Logon, enter_date)
    myassetimmo.Sno = Convert.ToInt32(strkey)
    myassetimmo.decl_year = Session("DECLYEAR").ToString
    myassetimmo.updateMyimmovableAssetsDetails()
    dtgriddisp.EditItemIndex = -1
    Loadmyasset_immovable()
End Sub

Private Sub dtgriddisp_DeleteCommand(ByVal source As System.Object, ByVal e As System.Web.UI.WebControls.DataGridCommandEventArgs) Handles dtgriddisp.DeleteCommand
    If e.CommandName = "Delete" Then
        Dim STRKEY As String = CType(e.Item.FindControl("Label22"), Label).Text.ToString
        viewstate("strkey") = STRKEY
        Dim Result As DataSet
        Result = SqlHelper.ExecuteDataset(ConnectionString, "Usp_AR_GETMyimmovableAssetsDetailsBySno", STRKEY)
Dim stat As Char = Char.Parse(Result.Tables(0).Rows(0)("decl_status").ToString())
If stat <> "2" Then
    Dim Result1 As Integer
    Result1 = Convert.ToInt32(SqlHelper.ExecuteScalar(ConnectionString, "Usp_AR_DeleteMyDetails", STRKEY, "))
ElseIf stat = "2" Then
    Panel2.Visible = True
End If
Loadmyasset_immovable()
End If
End Sub
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    Dim Result As Integer
    Result = Convert.ToInt32(SqlHelper.ExecuteScalar(ConnectionString, "Usp_AR_DeleteMyDetails", viewstate("strkey").ToString, TextBox7.Text.ToString))
    Loadmyasset_immovable()
    TextBox7.Text = ""
    Panel2.Visible = False
End Sub
Private Sub dtgridinput_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles dtgridinput.SelectedIndexChanged
End Sub
End Class

View_asset_immovable.aspx.vb
Imports assetdeclaration.AssetBusinessLogic
Public Class View_immovable
    Inherits System.Web.UI.Page
#Region " Web Form Designer Generated Code "
    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
End Sub
Protected WithEvents Label8 As System.Web.UI.WebControls.Label
Protected WithEvents HyperLink3 As System.Web.UI.WebControls.HyperLink
Protected WithEvents Label9 As System.Web.UI.WebControls.Label
Protected WithEvents Label10 As System.Web.UI.WebControls.Label
Protected WithEvents Label12 As System.Web.UI.WebControls.Label
Protected WithEvents lbl_name As System.Web.UI.WebControls.Label
Protected WithEvents Label11 As System.Web.UI.WebControls.Label
Protected WithEvents lbl_desig As System.Web.UI.WebControls.Label
Protected WithEvents Label13 As System.Web.UI.WebControls.Label
Protected WithEvents lbl_doj As System.Web.UI.WebControls.Label
Protected WithEvents Label14 As System.Web.UI.WebControls.Label
Protected WithEvents lbl_basicpay As System.Web.UI.WebControls.Label
Protected WithEvents Label15 As System.Web.UI.WebControls.Label
Protected WithEvents dtgriddisp As System.Web.UI.WebControls.DataGrid

'NOTE: The following placeholder declaration is required by the Web Form Designer.
'Do not delete or move it.
Private designerPlaceholderDeclaration As System.Object

Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Init
'CODEGEN: This method call is required by the Web Form Designer
'Do not modify it using the code editor.
InitializeComponent()
End Sub

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
'Put user code to initialize the page here
If Not Page.IsPostBack Then
Loadmyassest_immovable()
End If
End Sub

Private Sub Loadmyassest_immovable()
Dim ObjStaff As New Staff.StaffDetails
Dim Logon As String = Request.ServerVariables("Logon_user")
Dim Staffno As Integer = ObjStaff.GetStaffNumber(Logon)
Dim DsStaffInfo As DataSet = ObjStaff.GetBasicData(Staffno)
lbl_name.Text = DsStaffInfo.Tables(0).Rows(0).Item("Staff_nm").ToString
lbl_desig.Text = DsStaffInfo.Tables(0).Rows(0).Item("Desig").ToString
lbl_doj.Text = Format(DsStaffInfo.Tables(0).Rows(0).Item("Doj"), "dd/MMM/yyyy")
Dim dsData1 As DataSet = ObjStaff.GetPaymentDetails(Staffno)
lbl_basicpay.Text = CStr(dsData1.Tables(0).Rows(0).Item("amount"))
Dim mydispassestimmovable As New asset_immovable(Staffno)
mydispassestimmovable.decl_year = Request.Params("ddlcaseyear")
dtgriddisp.DataSource = mydispassestimmovable.GetMyimmovableAssetsDetails
dtgriddisp.DataBind()
dtgriddisp.Columns(0).Visible = False
End Sub
End Class

ViewallStaffDeclarations.aspx.vb
Imports assetdeclaration.AssetBusinessLogic
Public Class ViewAllStaffDeclarations
Inherits System.Web.UI.Page

#Region "Web Form Designer Generated Code"

'This call is required by the Web Form Designer.
<System.Diagnostics.DebuggerStepThrough()>
Private Sub InitializeComponent()
End Sub

Protected WithEvents Label8 As System.Web.UI.WebControls.Label
Protected WithEvents HyperLink1 As System.Web.UI.WebControls.HyperLink
Protected WithEvents Label9 As System.Web.UI.WebControls.Label
Protected WithEvents Label15 As System.Web.UI.WebControls.Label
Protected WithEvents dtgriddisp As System.Web.UI.WebControls.DataGrid

'NOTE: The following placeholder declaration is required by the Web Form Designer.
'Do not delete or move it.
Private designerPlaceholderDeclaration As System.Object

Private Sub Page_Init(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Init
'CODEGEN: This method call is required by the Web Form Designer
'Do not modify it using the code editor.
InitializeComponent()
End Sub
#End Region

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

'Put user code to initialize the page here
If Not Page.IsPostBack Then
LoadAllStaffassett()
End If
End Sub

Private Sub LoadAllStaffassett()
Dim ObjStaff As New Staff.StaffDetails
Dim Logon As String = Request.ServerVariables("Logon_user")
Dim Staffno As Integer = ObjStaff.GetStaffNumber(Logon)
Dim mydispassestmovable As New StaffDeclaration
'mydispassestmovable.decl_year = Request.Params("ddlcaseyear")
dtgriddisp.DataSource = mydispassestmovable.GetMyStaffDeclarationDetails(Staffno)
dtgriddisp.DataBind()
End Sub

End Class

Web.Config
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.web>
    <!-- DYNAMIC DEBUG COMPILATION
    Set compilation debug="true" to insert debugging symbols (.pdb information)
    into the compiled page. Because this creates a larger file
    that executes
    more slowly, you should set this value to true only when
    debugging and to
    false at all other times. For more information, refer to the
    documentation about
    debugging ASP.NET files. -->
    <compilation defaultLanguage="vb" debug="true"/>
    <!-- CUSTOM ERROR MESSAGES
    Set customErrors mode="On" or "RemoteOnly" to enable custom
    error messages, "Off" to disable.
    Add <error> tags for each of the errors you want to handle.
    "On" Always display custom (friendly) messages.
    "Off" Always display detailed ASP.NET error information.
    "RemoteOnly" Display custom (friendly) messages only to users
    not running
    on the local Web server. This setting is recommended for
    security purposes, so
    that you do not display application detail information to
    remote clients.'RemoteOnly
    -->
    <customErrors mode="Off"/>
    <!-- AUTHENTICATION
    This section sets the authentication policies of the
    application. Possible modes are "Windows",
    "Forms", "Passport" and "None"
    "None" No authentication is performed.
    "Windows" IIS performs authentication (Basic, Digest, or
    Integrated Windows) according to
    its settings for the application. Anonymous access must be
    disabled in IIS.
    "Forms" You provide a custom form (Web page) for users to
    enter their credentials, and then
    you authenticate them in your application. A user credential
    token is stored in a cookie.
    "Passport" Authentication is performed via a centralized
    authentication service provided
    by Microsoft that offers a single logon and core profile
    services for member sites.
    -->
    <authentication mode="Windows"/>
    <identity impersonate="true"
      userName="registry:HKLM\Software\ASPNET_SETREG,userName"
      password="registry:HKLM\Software\ASPNET_SETREG,password"></identity>
    <!-- AUTHORIZATION
This section sets the authorization policies of the application. You can allow or deny access to application resources by user or role. Wildcards: "*" means everyone, "?" means anonymous (unauthenticated) users.

```xml
<authorization>
  <allow users="*"/> <!-- Allow all users -->
  <!-- allow users="[comma separated list of users]" roles="[comma separated list of roles]"/ -->
  <deny users="[comma separated list of users]" roles="[comma separated list of roles]"/> <!--
</authorization>
```

```xml
<-- APPLICATION-LEVEL TRACE LOGGING
Application-level tracing enables trace log output for every page within an application.
Set trace enabled="true" to enable application trace logging.
If pageOutput="true", the trace information will be displayed at the bottom of each page. Otherwise, you can view the application trace log by browsing the "trace.axd" page from your web application root.

<!-- trace enabled="false" requestLimit="10" pageOutput="false" traceMode="SortByTime" localOnly="true"/>
```

```xml
<-- SESSION STATE SETTINGS
By default ASP.NET uses cookies to identify which requests belong to a particular session.
If cookies are not available, a session can be tracked by adding a session identifier to the URL.
To disable cookies, set sessionState cookieless="true".

<!--
sessionState mode="InProc" stateConnectionString="tcpip=127.0.0.1:42424" sqlConnectionString="data source=127.0.0.1;Trusted_Connection=yes" cookieless="false" timeout="20"/>
```

```xml
<-- GLOBALIZATION
This section sets the globalization settings of _u104 ?e application.

<!--
globalization requestEncoding="utf-8" responseEncoding="utf-8"/>
```

```xml
</system.web>
```

```xml
<appSettings><add key="assetdeclaration.Staff.StaffDetails" value="http://localhost/StaffService/StaffDetails.asmx"/>
</appSettings>
</configuration>
```