DOSSIER DE DEVELOPPEMENT DU PROJET Provision_D

18/02/2021

Partie 1

Projet

Projet

Code

Initialisation de Provision_D

```
// Les variables globales au projet (public)
Vpj_AfficheID          est un booléen
Vpj_Indice             est un entier
Vpj_MessageSupprime    est un booléen
Vpj_AfficheID          = Faux
Vpj_MessageSupprime    = Vrai
```

Partie 2

Fenêtre WINDEV

FEN_Compteur_liste

Code

Déclarations globales de FEN_Compteur_liste

```
PROCÉDURE MaFenêtre()
// déclaration des variables dispo partout dans la fenêtre car on est dans la declaration globale
Vl_compteur  est une entier sur 8 octets
Vl_action    est une chaîne
```

Fin d'initialisation de FEN_Compteur_liste

```
TA_Compteur.COL_CPT_ID..Visible=Vpj_AfficheID
// la procedure permet d'afficher la liste des enregistrements
pl_Afficheliste(Vl_compteur)
DonneFocus(TA_Compteur)
```

FEN_Compteur_liste

Code des champs

Clic sur BTN_Afficher

```
SI TA_Compteur.COL_CPT_ID <> 0 ALORS
    Ouvre(FEN_Compteur_fiche,Vl_compteur,"AFFI")
    // la procedure permet d'afficher la liste des enregistrements
    pl_Afficheliste(Vl_compteur)
    DonneFocus(TA_Compteur)
SINON
    Erreur("Il n'y a pas d'enregistrement sélectionné, Affichage impossible.")
FIN
```

Clic sur BTN_Modifier

```
SI TA_Compteur.COL_CPT_ID <> 0 ALORS
    Ouvre(FEN_Compteur_fiche,Vl_compteur,"MODI")
    pl_Afficheliste(Vl_compteur)
    DonneFocus(TA_Compteur)
SINON
    Erreur("Il n'y a pas d'enregistrement sélectionné, Modification impossible.")
FIN
```

Clic sur BTN_Nouveau

```
// ouvre la fenêtre compteur fiche
Ouvre(FEN_Compteur_fiche,Vl_compteur,"NOUV")
// la procedure permet d'afficher la liste des enregistrements
pl_Affiche liste(Vl_compteur)
// Donne le focus à un champ d'une fenêtre, d'une page  à une fenêtre

DonneFocus(TA_Compteur)
```

Clic sur BTN_Supprimer

```
SI TA_Compteur.COL_CPT_ID <> 0 ALORS
    Ouvre(FEN_Compteur_fiche, Vl_compteur,"SUPP")
    // la procedure permet d'afficher la liste des enregistrements
    pl_Affiche liste(Vl_compteur)
    DonneFocus(TA_Compteur)
SINON
    Erreur("Il n'y a pas d'enregistrement sélectionné, Suppression impossible.")
FIN
```

Sélection d'une ligne de TA_Compteur

```
Vl_compteur  = TA_Compteur.COL_CPT_ID
Vpj_Indice   = TableSelect(TA_Compteur)
```

Bouton gauche double-clic (WM_LBUTTONDBLCLK) sur TA_Compteur

```
Vl_compteur  = TA_Compteur.COL_CPT_ID
Vpj_Indice   = TableSelect(TA_Compteur)
SI Vl_action="CHOI" ALORS
    Ferme()
FIN
```

FEN_Compteur_liste

Procédures

Procédure locale pl_Affiche liste

```
PROCÉDURE pl_Affiche liste(v_compteur)
// Supprime toutes les lignes dans : •un champ Table mémoire, un champ Table hiérarchique mémoire, une
table affichée dans un champ Combo.


TableSupprimeTout(TA_Compteur)
//
HLibèreRequête(REQ_Compteur_liste)

REQ_Compteur_liste.P_CP_Actif=Null
// Déclare une requête au moteur HFSQL et exécute cette requête
HExécuteRequête(REQ_Compteur_liste)
SI ErreurDétectée() ALORS Erreur() ; RETOUR
// Affichage
FichierVersTableMémoire(TA_Compteur,REQ_Compteur_liste)
```

```
// // Dé-sélectionne les lignes dans le champ Table "TA_Compteur"
TableSelectMoins(TA_Compteur)
// si indice =0 alors retour
SI Vpj_Indice=0 ALORS
    RETOUR
SINON
    // Se place dans le tableau sur la valeur
    POUR TOUTE LIGNE DE TA_Compteur
        SI TA_Compteur.COL_CPT_ID=v_compteur ALORS
            TableSelectPlus(TA_Compteur,TA_Compteur)
        FIN
    FIN
    // si aucune sélection, se place sur le 1
    SI HNbEnr(REQ_Compteur_liste) > 0 ALORS
        SI TableSelect(TA_Compteur) < 1 ALORS
            TableSelectPlus(TA_Compteur,1)
            v_compteur=TA_Compteur.COL_CPT_ID
        FIN
    FIN

FIN
```

FEN_Compteur_fiche

Code

Déclarations globales de FEN_Compteur_fiche

```
PROCÉDURE MaFenêtre(LOCAL vl_Compteur ,LOCAL Vl_action)
HSurErreur(Compteur,hErrBlocage,pg_bloquer)
vl_NumEnr    est un entier
Vl_blocage   est une chaîne
```

Fin d'initialisation de FEN_Compteur_fiche

```
SAI_CPT_ID..Visible=Vpj_AfficheID

// En création
SI Vl_action="NOUV" ALORS
    FEN_Compteur_fiche..Titre="Nouveau Compteur"
    //SC_adm..Etat=AffichageSeulement
    HRAZ(Compteur)
    FichierVersEcran()
FIN

// En modification ou suppression
SI Vl_action="MODI" OU Vl_action="SUPP" ALORS
    // lecture de l'enregistrement et recherche son No d'enregistrement
    HLitRecherchePremier(Compteur,CPT_ID,vl_Compteur)
    vl_NumEnr    = HNumEnr(Compteur)
    // test de blocage
    Vl_blocage   = HInfoBlocage(Compteur,vl_NumEnr)
    SI pg_locked(Vl_blocage) ALORS
        Ferme()
    SINON
        // pas de blocage
        // je le lis et je le bloque
        HLitRecherchePremier(Compteur,CPT_ID,vl_Compteur,hBlocageEcriture)
        SI HTrouve(Compteur) ALORS
            SI Vl_action="MODI" ALORS
                FEN_Compteur_fiche..Titre="Modifie le Compteur"
            FIN
            SI Vl_action="SUPP" ALORS
                FEN_Compteur_fiche..Titre = "Supprime le Compteur"
                GR_Affiche..Etat          = AffichageSeulement
            FIN
            // Initialise automatiquement les champs d'une fenêtre ou d'une page avec :•les valeurs
            des rubriques associées dans l'enregistrement en cours (chargé en mémoire) du fichier de
            données

            FichierVersEcran()
        FIN
    FIN
FIN

// En consultation
SI Vl_action="AFFI" ALORS
    FEN_Compteur_fiche..Titre="Affiche le Compteur"
```

```
    // lecture de l'enregistrement et recherche son No d'enregsitrement
    HLitRecherchePremier(Compteur,CPT_ID,vl_Compteur)
    SI PAS HTrouve(Compteur) ALORS
        Erreur(ErreurInfo())
        Ferme()
    SINON
        FichierVersEcran()
        GR_Affiche..Etat=AffichageSeulement
    FIN
FIN
```

FEN_Compteur_fiche

Code des champs

Clic sur BTN_Annuler

```
SI Vl_action="MODI" OU Vl_action="SUPP" ALORS
    // Débloque un enregistrement précédemment bloqué
    HDébloqueNumEnr(Compteur)
FIN
```

Clic sur BTN_Valider

```
Vl_MotifID       est un entier sur 8 octets    = 0
Vl_MotifLibelle  est une chaîne                = ""

SI Vl_action="NOUV" OU Vl_action="MODI" ALORS
    // Le nom Compteur ne peut pas etre vide
    SI SansEspace(SAI_CPT_Compteur) = "" ALORS
        Erreur("Le nom du Compteur doit être renseigné.","Ressaisir !")
        DonneFocus(SAI_CPT_Compteur)
        RETOUR
    FIN
    // vérification intégrité
    HLibèreRequête(REQ_Compteur_VI)
    REQ_Compteur_VI.P_CPT_ID      = SAI_CPT_ID
    REQ_Compteur_VI.P_CPT_Compteur= SAI_CPT_Compteur
    SI PAS HExécuteRequête(REQ_Compteur_VI,hRequêteDéfaut) ALORS
        RETOUR
    SINON
        // Positionne sur le premier enregistrement d'un fichier de données en fonction d'une rubrique
        de parcours. L'enregistrement est lu et les variables HFSQL (par exemple Client.Nom,

        // c'est-à-dire la rubrique Nom du fichier Client) sont mises à jour.
        HLitPremier(REQ_Compteur_VI)
        // Renvoie le nombre d'enregistrements d'un fichier de données, d'une requête ou d'une vue
        HFSQL : enregistrements actifs, rayés, supprimés, etc.

        SI HNbEnr(REQ_Compteur_VI) > 0 ALORS
            Erreur("L' information " + SAI_CPT_Compteur + " existe déjà, ressaisir")
            DonneFocus(SAI_CPT_Compteur)
            RETOUR
```

```
        SINON
            EcranVersFichier()
            SI Vl_action="NOUV" ALORS
                HAjoute(Compteur)
            FIN
            SI Vl_action="MODI" ALORS
                ////             SI Vpj_MotifEnModif=Vrai ALORS
                ////                 TANTQUE Vl_MotifID=0
                ////
                (Vl_MotifID,Vl_MotifLibelle) =
                OuvrePopupPosition(FEN_PPMotif_fiche,poSelonChamp,SAI_CPT_Compteur,"",Vl_action)

                ////                     SI Vl_MotifID=0 ALORS
                ////
                Erreur("Le motif est obligatoire","Sélectionner à nouveau.")
                ////                     FIN
                ////                 FIN
                ////             FIN
                ////
                pg_traceG("Compteur",Compteur.CPT_ID,Compteur.CPT_Compteur,"",0,"",Vl_action,"",Vl_Mot
                ifID,Vl_MotifLibelle)

                HModifie(Compteur)
                HDébloqueNumEnr(Compteur,hNumEnrEnCours)
            FIN
            vl_Compteur=Compteur.CPT_ID
        FIN
    FIN
FIN



SI Vl_action="SUPP" ALORS
    Vl_nbo   est un entier    = 0
    //  vl_key   est une chaine
    // supprime enregistrement
    SI OuiNon(Non,"Confirmez la suppression du Compteur affichée  " + SAI_CPT_Compteur) = Oui ALORS
        // Vérification dans CatCompteur
        ////     vl_key=HFiltreIdentique(CatCompteur,CCP_CPT_ID,vl_Compteur)
        ////     //
        ////     SI vl_key<>"" ALORS
        ////         HLitPremier(CatCompteur,vl_key)
        ////         SI PAS HEnDehors(CatCompteur) ALORS
        ////             Vl_nbo=1
        ////         SINON
        ////             Vl_nbo=0
        ////         FIN
        ////     FIN
        ////     HDésactiveFiltre(CatCompteur)

        SI Vl_nbo = 0 ALORS
            ////         SI Vpj_MotifEnSuppression=Vrai ALORS
            ////             TANTQUE Vl_MotifID=0
            ////
            (Vl_MotifID,Vl_MotifLibelle) =
            OuvrePopupPosition(FEN_PPMotif_fiche,poSelonChamp,SAI_CPT_Compteur,"",Vl_action)

            ////                 SI Vl_MotifID=0 ALORS
            ////                     Erreur("Le motif est obligatoire","Sélectionner à nouveau.")
            ////                 FIN
            ////             FIN
            ////         FIN
            ////
            pg_traceG("Compteur",Compteur.CPT_ID,Compteur.CPT_Compteur,"",0,"",Vl_action,"",Vl_MotifID
            ,Vl_MotifLibelle)
```

```
            HSupprime(Compteur)
            vl_Compteur=0
            SI Vpj_MessageSupprime=Vrai ALORS
                Info("Le Compteur "+SAI_CPT_Compteur+" est supprimée.")
            FIN
        SINON
            Erreur("Le Compteur "+SAI_CPT_Compteur+" est utilisée. Suppression impossible." )

        FIN
    FIN
FIN

Ferme()
```

FEN_Societe_liste

Code

Déclarations globales de FEN_Societe_liste

```
PROCÉDURE MaFenêtre()
Vl_societe    est un entier sur 8 octets
Vl_action     est une chaîne
```

Fin d'initialisation de FEN_Societe_liste

```
// La propriété Visible permet de gérer la visibilité d'un élément. Cette propriété est utilisable sur
: •les champs d'une fenêtre, d'une page ou d'un état


TA_Societe.COL_SO_ID..Visible=Vpj_AfficheID
pl_Afficheliste(Vl_societe)
DonneFocus(TA_Societe)
```

FEN_Societe_liste

Code des champs

Clic sur BTN_Afficher

```
SI TA_Societe.COL_SO_ID <> 0 ALORS
    Ouvre(FEN_Societe_fiche,Vl_societe,"AFFI")
    pl_Afficheliste(Vl_societe)
    DonneFocus(TA_Societe)
SINON
    Erreur("Il n'y a pas d'enregistrement sélectionné, Affichage impossible.")
FIN
```

Clic sur BTN_Ajouter

```
Ouvre(FEN_Societe_fiche,Vl_societe,"NOUV")
pl_Afficheliste(Vl_societe)
DonneFocus(TA_Societe)
```

Clic sur BTN_Modifier

```
SI TA_Societe.COL_SO_ID <> 0 ALORS
    Ouvre(FEN_Societe_fiche,Vl_societe,"MODI")
    pl_Afficheliste(Vl_societe)
```

```
    DonneFocus(TA_Societe)
SINON
    Erreur("Il n'y a pas d'enregistrement sélectionné, Modification impossible.")
FIN
```

Clic sur BTN_Supprimer

```
SI TA_Societe.COL_SO_ID<> 0 ALORS
    Ouvre(FEN_Societe_fiche, Vl_societe,"SUPP")
    pl_Afficheliste(Vl_societe)
    DonneFocus(TA_Societe)
SINON
    Erreur("Il n'y a pas d'enregistrement sélectionné, Suppression impossible.")
FIN
```

Sélection d'une ligne de TA_Societe

```
Vl_societe   = TA_Societe.COL_SO_ID
// indice de la ligne sélectionne
Vpj_Indice   = TableSelect(TA_Societe)
```

Bouton gauche double-clic (WM_LBUTTONDBLCLK) sur TA_Societe

```
Vl_societe   = TA_Societe.COL_SO_ID
Vpj_Indice   = TableSelect(TA_Societe)
SI Vl_action="CHOI" ALORS
    Ferme()
FIN
```

FEN_Societe_liste

Procédures

Procédure locale pl_Afficheliste

```
PROCÉDURE pl_Afficheliste(v_societe)
// RAZ supprime contenu entier de la table
TableSupprimeTout(TA_Societe)
// Libère les ressources d'une requête
HLibèreRequête(REQ_Societe_liste)
//
REQ_Societe_liste.P_SOC_Actif=Null
//// Initialise la requête
HExécuteRequête(REQ_Societe_liste)
SI ErreurDétectée() ALORS Erreur() ; RETOUR

// Remplit un champ de type "Table mémoire" avec tous les enregistrements d'un fichier de données,
d'une vue HFSQL ou d'une requête

FichierVersTableMémoire(TA_Societe,REQ_Societe_liste)
// // Dé-sélectionne toutes les lignes  dans le champ Table "TA_SOCIETE"
TableSelectMoins(TA_Societe)
```

```
SI Vpj_Indice=0 ALORS
    RETOUR
SINON
    // Se place dans le tableau sur la valeur
    POUR TOUTE LIGNE DE TA_Societe
        SI TA_Societe.COL_SO_ID=v_societe ALORS
            TableSelectPlus(TA_Societe,TA_Societe)
        FIN
    FIN
    // si aucune sélection, se place sur le 1
    SI HNbEnr(REQ_Societe_liste) > 0 ALORS
        SI TableSelect(TA_Societe) < 1 ALORS
            TableSelectPlus(TA_Societe,1)
            v_societe=TA_Societe.COL_SO_ID
        FIN
    FIN

FIN
```

FEN_Societe_fiche

Code

Déclarations globales de FEN_Societe_fiche

```
PROCÉDURE MaFenêtre(LOCAL Vl_Societe ,LOCAL Vl_action)
HSurErreur(Société,hErrBlocage,pg_bloquer)
vl_NumEnr         est un entier
sVl_blocage       est une chaîne
```

Fin d'initialisation de FEN_Societe_fiche

```
// affiche id de la Societe a vrai
SAI_SOC_ID..Visible=Vpj_AfficheID

// En création
SI Vl_action="NOUV" ALORS
    FEN_Societe_fiche..Titre="Nouveau Societe"
    //  SC.adm..Etat=AffichageSeulement
    HRAZ(Société)
    FichierVersEcran()
FIN

// En modification ou suppression
SI Vl_action="MODI" OU Vl_action="SUPP" ALORS
    // lecture de l'enregistrement et recherche son No d'enregistrement


    HLitRecherchePremier(Société,SOC_ID,Vl_Societe)


    // Renvoie le numéro de l'enregistrement en cours dans le fichier de données HFSQL
    vl_NumEnr    = HNumEnr(Société)
    // test de blocage
    sVl_blocage  = HInfoBlocage(Société,vl_NumEnr)
    SI pg_locked(sVl_blocage) ALORS
        Ferme()
    SINON
        // pas de blocage
        // je le lis et je le bloque
        HLitRecherchePremier(Société,SOC_ID,Vl_Societe,hBlocageEcriture)
        SI HTrouve(Société) ALORS
            SI Vl_action="MODI" ALORS
                FEN_Societe_fiche..Titre="Modifie la Societe"
            FIN
            SI Vl_action="SUPP" ALORS
                FEN_Societe_fiche..Titre  = "Supprime la societe "
                GR_Affiche..Etat          = AffichageSeulement
            FIN
            FichierVersEcran()
        FIN
    FIN
FIN

// En consultation
SI Vl_action="AFFI" ALORS
    // le nom de la fenêtre devient affiche la societe
```

```
    FEN_Societe_fiche..Titre="Affiche la Societe "
    // lecture de l'enregistrement et recherche son No d'enregistrement
    HLitRecherchePremier(Société,SOC_ID,Vl_Societe)
    // Vérifie si l'enregistrement en cours correspond au filtre ou à la recherche en cours.
    SI PAS HTrouve(Société) ALORS
        Erreur(ErreurInfo())
        Ferme()
    SINON
        // Initialise automatiquement les champs d'une fenêtre ou d'une page avec :•les valeurs des
        rubriques associées dans l'enregistrement en cours (chargé en mémoire) du fichier de données

        FichierVersEcran()
        // ici on a mis les zones dans un groupe pour mettre letat à affichage
        GR_Affiche..Etat=AffichageSeulement
    FIN
FIN
```

FEN_Societe_fiche

Code des champs

Clic sur BTN_Annuler

```
SI Vl_action="MODI" OU Vl_action="SUPP" ALORS
    // Débloque un enregistrement précédemment bloqué
    HDébloqueNumEnr(Société)
FIN
```

Clic sur BTN_Valider

```
Vl_MotifID       est un entier sur 8 octets     = 0
Vl_MotifLibelle  est une chaîne                 = ""


SI Vl_action="NOUV" OU Vl_action="MODI" ALORS
    // Le nom Societe ne peut pas être vide il faut le renseigner
    SI SansEspace(SAI_SOC_Nom) = "" ALORS
        Erreur("Le nom de la société doit être renseigné.","Ressaisir !")
        DonneFocus(SAI_SOC_Nom)
        RETOUR
    FIN
    // vérification intégrité
    // Libère les ressources d'une requête
    HLibèreRequête(REQ_Societe_VI2)

    REQ_Societe_VI2.P_SOC_ID  = SAI_SOC_ID
    REQ_Societe_VI2.P_SOC_Nom = SAI_SOC_Nom
    // Déclare une requête au moteur HFSQL et exécute cette requête. Cette requête peut correspondre à
    une requête créée sous l'éditeur de requêtes

    //une variable de type Requête SQL

    SI PAS HExécuteRequête(REQ_Societe_VI2,hRequêteDéfaut) ALORS
        RETOUR
    SINON
```

```
        HLitPremier(REQ_Societe_VI2)
        // Renvoie le nombre d'enregistrements d'un fichier de données, d'une requête ou d'une vue
        HFSQL : enregistrements actifs, rayés, supprimés, etc

        SI HNbEnr(REQ_Societe_VI2) > 0 ALORS
            Erreur("L' information " + SAI_SOC_Nom + " existe déjà, ressaisir")
            DonneFocus(SAI_SOC_Nom)
            RETOUR
        SINON
            EcranVersFichier()
            SI Vl_action="NOUV" ALORS
                // Ajoute l'enregistrement présent en mémoire dans le fichier de données
                HAjoute(Société)
            FIN
            SI Vl_action="MODI" ALORS
                ////             SI Vpj_MotifEnModif=Vrai ALORS
                ////                 TANTQUE Vl_MotifID=0
                ////
                (Vl_MotifID,Vl_MotifLibelle) =
                OuvrePopupPosition(FEN_PPMotif_fiche,poSelonChamp,SAI_CPT_Compteur,"",Vl_action)

                ////                     SI Vl_MotifID=0 ALORS
                ////
                Erreur("Le motif est obligatoire","Sélectionner à nouveau.")
                ////                     FIN
                ////                 FIN
                ////             FIN
                ////
                pg_traceG("Compteur",Compteur.CPT_ID,Compteur.CPT_Compteur,"",0,"",Vl_action,"",Vl_Mot
                ifID,Vl_MotifLibelle)



                // Modifie l'enregistrement spécifié ou l'enregistrement présent en mémoire dans le
                fichier de données (la requête ou la vue).

                HModifie(Société)
                // Débloque un enregistrement précédemment bloqué
                HDébloqueNumEnr(Société,hNumEnrEnCours)
            FIN
            Vl_Societe=Société.SOC_ID
        FIN
    FIN
FIN



SI Vl_action="SUPP" ALORS
    Vl_nbo   est un entier    = 0
    //  vl_key   est une chaine
    // supprime enregistrement
    SI OuiNon(Non,"Confirmez la suppression de la société affichée  " + SAI_SOC_Nom) = Oui ALORS
        // Vérification dans CatCompteur
        ////     vl_key=HFiltreIdentique(CatCompteur,CCP_CPT_ID,vl_Compteur)
        ////     //
        ////     SI vl_key<>"" ALORS
        ////         HLitPremier(CatCompteur,vl_key)
        ////         SI PAS HEnDehors(CatCompteur) ALORS
        ////             Vl_nbo=1
        ////         SINON
        ////             Vl_nbo=0
        ////         FIN
        ////     FIN
        ////     HDésactiveFiltre(CatCompteur)
        ////     //
```

```
        //
        SI Vl_nbo = 0 ALORS
            ////         SI Vpj_MotifEnSuppression=Vrai ALORS
            ////             TANTQUE Vl_MotifID=0
            ////
            (Vl_MotifID,Vl_MotifLibelle) =
            OuvrePopupPosition(FEN_PPMotif_fiche,poSelonChamp,SAI_CPT_Compteur,"",Vl_action)

            ////                 SI Vl_MotifID=0 ALORS
            ////                     Erreur("Le motif est obligatoire","Sélectionner à nouveau.")
            ////                 FIN
            ////             FIN
            ////         FIN
            ////
            pg_traceG("Compteur",Compteur.CPT_ID,Compteur.CPT_Compteur,"",0,"",Vl_action,"",Vl_MotifID
            ,Vl_MotifLibelle)

            HSupprime(Société)
            Vl_Societe=0
            SI Vpj_MessageSupprime=Vrai ALORS
                Info("La société "+SAI_SOC_Nom+" est supprimée.")
            FIN
        SINON
            Erreur("La société  "+SAI_SOC_Nom+" est utilisée. Suppression impossible." )

        FIN
    FIN
FIN

Ferme()
```

FEN_Combo

Code

Déclarations globales de FEN_Combo

```
PROCÉDURE MaFenêtre()
// variable id_societe de type numerique
Vl_id_soc est un entier
```

Fin d'initialisation de FEN_Combo

```
//La propriété Visible permet de gérer la visibilité d'un élément. Cette propriété est utilisable sur :
•les champs d'une fenêtre, d'une page ou d'un état

// ici on ne rend pas visible l'id de la societe dans le combo
COMBO.COL_SOC_ID..Visible=Faux
```

FEN_Combo

Code des champs

Initialisation de CMB_SOCIETE_PP

```
// condition en plus : sélectionne que les actif
// Positionne sur le premier enregistrement d'un fichier de données en fonction d'une rubrique de
parcours.

HLitPremier(Société,SOC_Nom)
// Vérifie si l'enregistrement en cours correspond au filtre ou à la recherche en cours.
TANTQUE HTrouve(Société)
    SI Société.SOC_Actif=Vrai ALORS

        // Ajoute une ligne dans le champ Table,

        TableAjoute(CMB_SOCIETE_PP,Société.SOC_ID+ TAB + Société.SOC_Nom)
    FIN

    HLitSuivant(Société,SOC_Nom)

FIN
```

Fin d'initialisation de COMBO

```
// ajout de 2 lignes ( tous et aucun dans le combo ils auront un id specifique )
TableAjoute(COMBO,-1 + TAB + "tous")
TableAjoute(COMBO,-2 + TAB+ "Aucun")
```

Sélection d'une ligne de COMBO

```
// affichage de l'id
Vl_id_soc             = COMBO.COL_SOC_ID
// le sai_texte prend la valeur de l'id de la Societe
SAI_Texte..Valeur= Vl_id_soc
```

FEN_Excel

Code

Déclarations globales de FEN_Excel

```
PROCÉDURE MaFenêtre()
gbModif                est un booléen
gMaFeuille             est un xlsDocument
gbDocumentOuvert  est un booléen
gnNumLigne             est un entier
gnNumColonne      est un entier
gnFeuilleCourante est un entier
gsSchErreur            est une chaîne    = ""
gnNBCOL                est un entier
y                      est un xlsDocument
```

FEN_Excel

Code des champs

Clic sur BTN_Fichier

```
sFichier est une chaîne

//Ouvre le sélecteur de fichiers du système en cours
sFichier = fSélecteur("C:\Users\pc-hugo\Desktop\Nouveau dossier", "activite.xlsx", ...
"Sélectionner un fichier", "Fichier Excel (*.XLSX)" + TAB + ...
"*.xlsx" + RC + "Tous fichiers (*.*)" + TAB + "*.*", "XLSX", fselOuvre + fselExiste)

// si la chaine est différent de vide alors cette chaine prend la valeur de sai_fichier
SI sFichier <> "" ALORS
    SAI_Fichier= sFichier
SINON
    Ferme()
FIN

// si il n'y a pas de fichier sélectionne alors vérifiez vos informations s'il vous plait
SI SAI_Fichier="" ALORS
    Info("vérifiez vos informations s'il vous plait ")
SINON
    // Ferme un fichier XLS
    xlsFerme(y)
    // Ouvre un fichier Excel (fichiers xls ou xlsx)
    y=xlsOuvre(SAI_Fichier,xlsEcriture)
    // ErreurDétectée correspond à :
    //Faux si la dernière fonction WLangage utilisée a réussi.
    // vrai si la dernière fonction WLangage utilisée a provoqué une erreur non fatale.

    SI ErreurDétectée("", ErreurInfo) ALORS
        RETOUR
    FIN
FIN
```

Clic sur BTN_Supprimer

```
// Supprime toutes les lignes du fichier excel

i est un entier
POUR i=TABLE_Activite..Occurrence À 1 PAS -1
    TableSupprime(TABLE_Activite,i)
    SI ErreurDétectée ALORS
        Erreur("Impossible de supprimer l'enregistrement."+RC+HErreurInfo())
        RETOUR
    FIN
FIN
```

Clic sur BTN_VALIDER

```
nBrLigne est un entier =xlsNbLigne(y)
HSupprimeTout(Activite)
POUR i= 2 À nBrLigne
    SI SansEspace( xlsDonnée(y,i,1)) <> "" ALORS
        Activite.ACT_Section      = xlsDonnée(y ,i,1)
        Activite.ACT_Matricule        = xlsDonnée(y,i,2)
        Activite.ACT_Nom_Prenom       = xlsDonnée(y,i,3)
        Activite.ACT_RTT          = xlsDonnée(y,i,4)
        Activite.ACT_RTT_N_1      = xlsDonnée(y,i,5)
        Activite.ACT_RECM         = xlsDonnée(y,i,6)
        Activite.ACT_RECH         = xlsDonnée(y,i,7)
        Activite.ACT_RC               = xlsDonnée(y,i,8)
        Activite.ACT_CET_H            = xlsDonnée(y,i,9)
        Activite.ACT_CP_N         = xlsDonnée(y,i,10)
        Activite.ACT_CP_N_1           = xlsDonnée(y,i,11)
        Activite.ACT_ANCI         = xlsDonnée(y,i,12)
        Activite.ACT_Repos            = xlsDonnée(y,i,13)
        Activite.ACT_Repos_N_1        = xlsDonnée(y,i,14)
        Activite.ACT_CET_J            = xlsDonnée(y,i,15)
        Activite.ACT_Repos_AN_N_DU    = xlsDonnée(y,i,16)
        // conversion pour les heures ci dessous avec les nouveaux champs ajoute dans la table activite

        Activite.ACT_N_RTT            = pl_Convert_heure(Activite.ACT_RTT)
        Activite.ACT_N_RTT_N_1        = pl_Convert_heure(Activite.ACT_RTT_N_1)
        Activite.ACT_N_RECM           = pl_Convert_heure(Activite.ACT_RECM)
        Activite.ACT_N_RECH           = pl_Convert_heure(Activite.ACT_RECH)
        Activite.ACT_N_RC         = pl_Convert_heure(Activite.ACT_RC)
        Activite.ACT_N_CET_H      = pl_Convert_heure(Activite.ACT_CET_H)
        // conversion pour les jours ci dessous
        Activite.ACT_N_CP_N           = pl_Convert_jour(Activite.ACT_CP_N)
        Activite.ACT_N_CP_N_1     = pl_Convert_jour(Activite.ACT_CP_N_1)
        Activite.ACT_N_ANCI           = pl_Convert_jour(Activite.ACT_ANCI)
        Activite.ACT_N_Repos      = pl_Convert_jour(Activite.ACT_Repos)
        Activite.ACT_N_Repos_N_1  = pl_Convert_jour(Activite.ACT_Repos_N_1)
        Activite.ACT_N_CET_J      = pl_Convert_jour(Activite.ACT_CET_J)

        HAjoute(Activite)
    FIN

FIN

// Rafraîchit l'affichage d'un champ Table à partir d'une position donnée :un champ Table fichier : les
modifications effectuées sur le fichier de données lié sont répercutées dans le champ


TableAffiche(TABLE_Activite)
```

FEN_Excel

Procédures

Procédure locale pl_Convert_heure

```
PROCÉDURE pl_Convert_heure(v_Hrecue )

vl_Hrecue         est une chaîne
vl_Hfinale        est une chaîne
vl_Position       est un entier
// heures en chaine de caractere
vl_HeureC         est une chaîne
 // minutes en chaine de caractere
vl_MinuteC        est une chaîne
vl_taille         est un entier
// // minutes  en numerique
vl_MinuteN        est un entier
vl_HeureNRetour   est un numérique
vl_estNegatif     est un booléen    = Faux
vl_Hrecue    = v_Hrecue
// exemple ici dessous pour se reperer dans le code
// "-10h23"



// Renvoie une chaîne de caractères sans les espaces : •situés à gauche et à droite.
// situés à l'intérieur de la chaîne.
vl_Hrecue    = SansEspace(v_Hrecue)
//si il ya  un - alors on se positionne au caractere a la position 2 et donc la valeur est negative
 sinon positive
SI Gauche(vl_Hrecue,1)="-" ALORS
    vl_Hrecue         = Milieu(vl_Hrecue,2)
    vl_estNegatif     = Vrai
SINON
    vl_estNegatif = Faux
FIN
// ici on recherche le caractere h
vl_Position  = Position(vl_Hrecue,"h")


SI vl_Position=0 ALORS
    vl_Hfinale="0"
SINON
    // heure en chaine
    vl_HeureC    = Gauche(vl_Hrecue,vl_Position - 1)
    // renvoie la taille
    vl_taille    = Taille(vl_Hrecue)
    vl_MinuteC   = Milieu(vl_Hrecue,vl_Position + 1,vl_taille - vl_Position)
    //conversion en numerique ( *100/60)
    vl_MinuteN   = (Val(vl_MinuteC) * 100 ) / 60

    // si négatif on met un - devant la valeur  sinon on ne met rien
    SI vl_estNegatif=Vrai ALORS
        vl_Hfinale = "-"+ vl_HeureC + "." + NumériqueVersChaîne(vl_MinuteN)
    SINON
        vl_Hfinale =  vl_HeureC + "." + NumériqueVersChaîne(vl_MinuteN)
    FIN


FIN
// Renvoie la valeur numérique d'une chaîne de caractères.
vl_HeureNRetour=Val(vl_Hfinale)
```

```
RENVOYER vl_HeureNRetour
```

Procédure locale pl_Convert_jour

```
PROCÉDURE pl_Convert_jour(v_Jrecue )
//ce qu'on a l'écran ( jour recue)
vl_Jrecue                 est une chaîne
// variable de sortie a l'affichage
vl_Jfinale                est une chaîne
// position du pointeur
vl_Position               est un entier
//jour en type car
vl_JourC             est une chaîne
// après la virgule en type varchar
vl_PartieDecimalC    est une chaîne
// taille de la chaine
vl_taille                 est un entier
// apres la virgule en type numerique
vl_PartieDecimalN    est un entier
// jour de retour en numerique
vl_JourNRetour            est un numérique
// si valeur negatif
vl_estNegatif             est un booléen    = Faux

// la variable locale jour recue reçoit la variable mis en paramètre lors de l'appel de la procedure
vl_Jrecue    = v_Jrecue

// "-10j23" //
// jour recue reçoit la fonction sans_espace sur Jrecue pour eviter
 des erreurs lors de la compilation du programme
vl_Jrecue    = SansEspace(v_Jrecue)

// si premier caractere est un - alors on se positionne au 2 deuxieme caractere et donc c une valeur
negative sinon pas negatif
SI Gauche(vl_Jrecue,1)="-" ALORS
    vl_Jrecue        = Milieu(vl_Jrecue,2)
    vl_estNegatif    = Vrai
SINON
    vl_estNegatif = Faux
FIN
// recherche la position de la lettre j dans la  chaine de caractere
vl_Position  = Position(vl_Jrecue,"j")

// si position nulle alors jour_finale null
SI vl_Position=0 ALORS
    vl_Jfinale="0"
SINON
    vl_JourC         = Gauche(vl_Jrecue,vl_Position - 1)
    vl_taille            = Taille(vl_Jrecue)
    vl_PartieDecimalC= Milieu(vl_Jrecue,vl_Position + 1,vl_taille - vl_Position)
    // comme le chiffre apres la virgule ne change pas la partie numerique = partie décimal
    //  vl_PartieDecimalN= Val(vl_PartieDecimalC)

    // si valeur négative alors on met un - devant la valeur et on concatène avec le jour qui est en
    var char + les valeurs après la virgule

    SI vl_estNegatif=Vrai ALORS
        vl_Jfinale = "-"+ vl_JourC + "." + vl_PartieDecimalC
    SINON
        vl_Jfinale =  vl_JourC + "." + vl_PartieDecimalC
    FIN


FIN
// JourNRetour reçoit la variable jour finale qui renvoie la valeur numérique d'une chaîne de
caractères.
```

```
vl_JourNRetour=Val(vl_Jfinale)
//renvoie de jour n retour
RENVOYER vl_JourNRetour
```

FEN_CARACTERE

Code

Déclarations globales de FEN_CARACTERE

```
PROCÉDURE MaFenêtre()
```

FEN_CARACTERE

Code des champs

Clic sur BTN_DECROISSANT

```
MonMot       est une chaîne    // mot initial

Resultat est une chaîne                                    // résultat final du mot

vl_taille    est un entier
// on va chercher le mot que l'utilisateur a rentre dans la zone de saisie
MonMot       = SAI_Texte
vl_taille    = Taille(MonMot)

vl_carN, vl_CarNp1 sont des chaînes // variable au caractere n et n+1

POUR k= 1 _À_ vl_taille   // renvoie 6-1 donc 5

    POUR n= 1 _À_ vl_taille -1

        vl_carN      = Milieu(MonMot,n,1)
        vl_CarNp1    = Milieu(MonMot,n+1,1)
        SI vl_carN < vl_CarNp1 ALORS

            SI n > 0 ALORS
                MonMot= Gauche(MonMot,n-1)+vl_CarNp1+ vl_carN+ Milieu(MonMot,n+1+1,vl_taille-n+1+1)
            FIN


        SINON
            // ne rien faire

        FIN

    FIN

FIN
Resultat=MonMot
Info(Resultat) //affichage
```

Clic sur BTN_Ordre_croissant

```
MonMot       est une chaîne              // mot de départ en entrée qui va être trier a la fin
```

```
Resultat est une chaîne                // affichage a la sortie
vl_taille    est un entier
// on va chercher le mot que l'utilisateur a rentre dans la zone de saisie
MonMot       = SAI_Texte
// taille en entier
vl_taille    = Taille(MonMot) // renvoie taille du mot en numerique

vl_carN, vl_CarNp1 sont des chaînes // variable au caractere n et n+1

POUR k= 1 _À_ vl_taille

    POUR n= 1 _À_ vl_taille -1  // -1 car sinon on sort du tableau

        vl_carN      = Milieu(MonMot,n,1) // vl_CarN se place au caractere a la position n
        vl_CarNp1    = Milieu(MonMot,n+1,1) // vl_CarNp1 se place au caractere a la position n +1
        SI vl_carN > vl_CarNp1 ALORS  // si n > n+1 alors on fait le traitement

            SI n > 0 ALORS  // si n > 0
                //transformation ici
                MonMot= Gauche(MonMot,n-1)+vl_CarNp1 + vl_carN + Milieu(MonMot,n+1+1,vl_taille-n+1+1)
            FIN

        SINON
            // ne rien faire

        FIN


    FIN


FIN

Resultat=MonMot // on met le contenu de la variable MonMot dans Résultat
Info(Resultat) //affichage
```

FEN_Categorie_fiche

Code

Déclarations globales de FEN_Categorie_fiche

```
PROCÉDURE MaFenêtre(LOCAL Vl_Categorie ,LOCAL Vl_action)
HSurErreur(Categorie,hErrBlocage,pg_bloquer)
vl_NumEnr         est un entier
Vl_blocage        est une chaîne
```

Fin d'initialisation de FEN_Categorie_fiche

```
SAI_CAT_ID..Visible=Vpj_AfficheID

// En création
SI Vl_action="NOUV" ALORS
    FEN_Categorie_fiche..Titre="Nouvelle Catégorie" // le titre de la fenetre s'appelle nouvelle
    categorie
    //SC_adm..Etat=AffichageSeulement
    HRAZ(Categorie)
    // Initialise : une ou toutes les variables des rubriques d'un fichier de données avec leurs
    valeurs par défaut.


    FichierVersEcran()
    //Initialise automatiquement les champs d'une fenêtre ou d'une page avec :•les valeurs des
    rubriques associées dans l'enregistrement en cours


FIN

// En modification ou suppression
SI Vl_action="MODI" OU Vl_action="SUPP" ALORS
    // lecture de l'enregistrement et recherche son No d'enregistrement
    HLitRecherchePremier(Categorie,CAT_ID,Vl_Categorie)
    vl_NumEnr    = HNumEnr(Categorie)
    // test de blocage
    Vl_blocage   = HInfoBlocage(Categorie,vl_NumEnr)
    SI pg_locked(Vl_blocage) ALORS
        Ferme()
    SINON
        // pas de blocage
        // je le lis et je le bloque
        HLitRecherchePremier(Categorie,CAT_ID,Vl_Categorie,hBlocageEcriture)
        SI HTrouve(Categorie) ALORS
            SI Vl_action="MODI" ALORS
                FEN_Categorie_fiche..Titre="Modifie la categorie"
            FIN
            SI Vl_action="SUPP" ALORS
                FEN_Categorie_fiche..Titre    = "Supprime la categorie "
                GR_Affiche..Etat          = AffichageSeulement
            FIN
            FichierVersEcran()
        FIN
    FIN
FIN
```

```
// En consultation
SI Vl_action="AFFI" ALORS
    FEN_Categorie_fiche..Titre="Affiche la categorie "
    // lecture de l'enregistrement et recherche son No d'enregsitrement
    HLitRecherchePremier(Categorie,CAT_ID,Vl_Categorie)
    SI PAS HTrouve(Categorie) ALORS
        Erreur(ErreurInfo())
        Ferme()
    SINON
        FichierVersEcran()
        GR_Affiche..Etat=AffichageSeulement
    FIN
FIN
```

FEN_Categorie_fiche

Code des champs

Clic sur BTN_Annuler

```
SI Vl_action="MODI" OU Vl_action="SUPP" ALORS
    // Débloque un enregistrement précédemment bloqué
    HDébloqueNumEnr(Categorie)
FIN
```

Clic sur BTN_Valider

```
SI Vl_action="NOUV" OU Vl_action="MODI" ALORS
    // Renvoie une chaîne de caractères sans les espaces : •situés à gauche et à droite et situés à
    l'intérieur de la chaîne.

    SI SansEspace(SAI_CAT_Nom) = "" ALORS
        Erreur("Le nom de la catégorie doit être renseigné.","Ressaisir !")
        //Donne le focus au champ SAI_CAT_Nom
        DonneFocus(SAI_CAT_Nom)
        RETOUR
    FIN
    // Libère les ressources d'une requête
    HLibèreRequête(REQ_Categorie_VI)
    // parametre cat_id = sai_cat_id
    // parametre cat_nom = sai_cat_nom
    REQ_Categorie_VI.P_CAT_ID = SAI_CAT_ID
    REQ_Categorie_VI.P_CAT_Nom    = SAI_CAT_Nom

    SI PAS HExécuteRequête(REQ_Categorie_VI,hRequêteDéfaut) ALORS
        RETOUR
    SINON
        HLitPremier(REQ_Categorie_VI)
        SI HNbEnr(REQ_Categorie_VI) > 0 ALORS
            Erreur("L' information " + SAI_CAT_Nom + " existe déjà, ressaisir")
            DonneFocus(SAI_CAT_Nom)
            RETOUR
```

```
        SINON
            EcranVersFichier()
            SI Vl_action="NOUV" ALORS
                HAjoute(Categorie)
            FIN
            SI Vl_action="MODI" ALORS
                // Modifie l'enregistrement spécifié ou l'enregistrement présent en mémoire dans le
                fichier de données (la requête)

                HModifie(Categorie)
                //Débloque un enregistrement précédemment bloqué
                HDébloqueNumEnr(Categorie,hNumEnrEnCours)
            FIN
            Vl_Categorie=Categorie.CAT_ID
        FIN
    FIN
FIN



SI Vl_action="SUPP" ALORS
    Vl_nbo   est un entier    = 0
    // Affiche un message dans une boîte de dialogue standard proposant les réponses "Oui" et "Non" et
    renvoie la réponse de l'utilisateur.

    SI OuiNon(Non,"Confirmez la suppression de la catégorie affichée  " + SAI_CAT_Nom) = Oui ALORS

        SI Vl_nbo = 0 ALORS
            HSupprime(Categorie)
            Vl_Categorie=0
            SI Vpj_MessageSupprime=Vrai ALORS
                Info("La catégorie "+SAI_CAT_Nom+" est supprimée.")
            FIN
        SINON
            Erreur("La catégorie  "+SAI_CAT_Nom+" est utilisée. Suppression impossible." )

        FIN
    FIN
FIN
// Ferme une fenêtre WINDEV (en renvoyant si nécessaire une valeur).
Ferme()
```

FEN_Categorie_liste

Code

Déclarations globales de FEN_Categorie_liste

```
PROCÉDURE MaFenêtre()
// déclaration des variables vl_categorie (entier) et action (chaine)
Vl_categorie est une entier sur 8 octets
Vl_action        est une chaîne
```

Fin d'initialisation de FEN_Categorie_liste

```
// fin d’initialisation on appelle la procedure afficheliste pour avoir les enregistrements
pl_Afficheliste(Vl_categorie)

TA_Categorie.COL_IDCategorie..Visible=Vpj_AfficheID
// Donne le focus au champ table
DonneFocus(TA_Categorie)
```

FEN_Categorie_liste

Code des champs

Clic sur BTN_Afficher

```
// si la colonne "id_categorie" du champ table TA_Categorie  est différent de zéro alors on ouvre la
fenêtre
categorie fiche pour afficher  les infos et on affiche la liste avec la procedure
// sinon si ce n'est pas différent de zéro alors il n'y a pas d'enregistrement sélectionné  affichage
impossible


SI TA_Categorie.COL_IDCategorie <> 0 ALORS
    Ouvre(FEN_Categorie_fiche,Vl_categorie,"AFFI")
    pl_Afficheliste(Vl_categorie)
    DonneFocus(TA_Categorie)
SINON
    Erreur("Il n'y a pas d'enregistrement sélectionné, Affichage impossible.")
FIN
```

Clic sur BTN_Ajouter

```
// on ouvre la fenêtre catégorie fiche ici il s'agit  d'un ajout
Ouvre(FEN_Categorie_fiche,Vl_categorie,"NOUV")
// on appelle la procedure afficheliste pour avoir les enregistrements
pl_Afficheliste(Vl_categorie)
DonneFocus(TA_Categorie)
```

Clic sur BTN_Modifier

```
// si la colonne "id_categorie" du champ table TA_Categorie  est différent de zero
 alors on ouvre la fenêtre categorie
 fiche pour modifier les infos et on affiche la liste avec la procedure
// sinon si ce n'est pas différent de zero
 alors il n'y a pas d'enregistrement sélectionné  Modification impossible
SI TA_Categorie.COL_IDCategorie <> 0 ALORS
    Ouvre(FEN_Categorie_fiche,Vl_categorie,"MODI")
    pl_Afficheliste(Vl_categorie)
    DonneFocus(TA_Categorie)
SINON
    Erreur("Il n'y a pas d'enregistrement sélectionné, Modification impossible.")
FIN
```

Clic sur BTN_Supprimer

```
// si la colonne "id"_categorie" du champ table TA_Categorie  est différent de zero
 alors on ouvre la fenêtre categorie
 fiche pour afficher  les infos et on affiche la liste avec la procedure
// sinon si ce n'est pas différent de zero
 alors il n'y a pas d'enregistrement sélectionné  suppression impossible
SI TA_Categorie.COL_IDCategorie<> 0 ALORS
    Ouvre(FEN_Categorie_fiche, Vl_categorie,"SUPP")
    pl_Afficheliste(Vl_categorie)
    DonneFocus(TA_Categorie)
SINON
    Erreur("Il n'y a pas d'enregistrement sélectionné, Suppression impossible.")
FIN
```

Sélection d'une ligne de TA_Categorie

```
Vl_categorie =  TA_Categorie.COL_IDCategorie
// Renvoie l'indice de l'élément sélectionné dans le champ Table
Vpj_Indice       = TableSelect(TA_Categorie)
```

FEN_Categorie_liste

Procédures

Procédure locale pl_Afficheliste

```
PROCÉDURE pl_Afficheliste(v_categorie)
// RAZ supprime contenu entier de la table
TableSupprimeTout(TA_Categorie)
// Libère les ressources d'une requête
HLibèreRequête(REQ_Categorie_liste)
// on va chercher dans la requête categorie liste les actifs qui sont a null
REQ_Categorie_liste.P_CAT_Actif=Null
//// Déclare une requête au moteur HFSQL et exécute cette requête
HExécuteRequête(REQ_Categorie_liste)
SI ErreurDétectée() ALORS Erreur() ; RETOUR

// Remplit un champ de type "Table mémoire" avec tous les enregistrements d'un fichier de données,
d'une vue HFSQL ou d'une requête

FichierVersTableMémoire(TA_Categorie,REQ_Categorie_liste)
// // Dé-sélectionne toutes les lignes  dans le champ Table "TA_categorie"
```

```
TableSelectMoins(TA_Categorie)

SI Vpj_Indice=0 ALORS
    RETOUR
SINON

    POUR TOUTE LIGNE DE TA_Categorie
        SI TA_Categorie.COL_IDCategorie=v_categorie ALORS
            TableSelectPlus(TA_Categorie,TA_Categorie)
        FIN
    FIN

    // Renvoie le nombre d'enregistrements d'un fichier de données, d'une requête ou d'une vue HFSQL :
    enregistrements actifs, rayés, supprimés, etc.

    SI HNbEnr(REQ_Categorie_liste) > 0 ALORS
        SI TableSelect(TA_Categorie) < 1 ALORS
            TableSelectPlus(TA_Categorie,1)
            v_categorie=TA_Categorie.COL_IDCategorie
        FIN
    FIN

FIN
```

FEN_SUITE

Code

Déclarations globales de FEN_SUITE

```
PROCÉDURE MaFenêtre()
Vl_compteur       est un entier sur 8 octets
Vl_action         est une chaîne
Vl_cat_compteur   est un entier sur 8 octets
```

Fin d'initialisation de FEN_SUITE

```
// cacher les colonnes CAC_CAT_ID CAC_CPT_ID, CAC_SOC_ID, CAC_ID avec la propriete visible

TA_Cat_Compteur.COL_CAC_CAT_ID..Visible    = Vpj_AfficheID
TA_Cat_Compteur.COL_CAC_CPT_ID..Visible    = Vpj_AfficheID
TA_Cat_Compteur.COL_CAC_SOC_ID..Visible    = Vpj_AfficheID
TA_Cat_Compteur.COL_CAC_ID..Visible        = Vpj_AfficheID

HLibèreRequête(REQ_Societe_liste)
REQ_Societe_liste.P_SOC_Actif=Vrai
TableAffiche(COMBO_Societe,taRéExécuteRequete)
TA_Compteur.COL_CPT_ID..Visible                = Vpj_AfficheID
TA_Cat_Compteur.COL_CAC_CAT_ID..Visible        = Vpj_AfficheID
pl_Afficheliste(Vl_cat_compteur)
DonneFocus(TA_Compteur)
```

FEN_SUITE

Code des champs

Clic sur BTN_Ajouter

```
// si l'utilisateur ne selectionne pas de Societe on a un message d'erreur
SI COMBO_Societe.COL_SO_ID =0 ALORS
    Erreur("Vous devez sélectionner une société.","Abandon")
    RETOUR
SINON
    // si l'utilisateur ne selectionne pas de categorie on a un message d'erreur
    SI COMBO_Categorie.COL_IDCategorie= 0 ALORS
        Erreur("Vous devez sélectionner une catégorie.","Abandon")
        RETOUR
    SINON

        // Initialise une ou toutes les variables des rubriques d'un fichier de données avec leurs
        valeurs par défaut.

        HRAZ(Cat_Compteur)

        Cat_Compteur.CAC_CPT_ID   = Vl_compteur
```

```
        Cat_Compteur.CAC_SOC_ID   = COMBO_Societe.COL_SO_ID
        Cat_Compteur.CAC_CAT_ID   = COMBO_Categorie.COL_IDCategorie
        // Positionne sur le premier enregistrement du fichier de données dont la valeur d'une rubrique
        spécifique est strictement égale à une valeur recherchée

        HLitRecherchePremier(Compteur,CPT_ID,Vl_compteur)


        SI HTrouve(Compteur) ALORS
            // on met la colonne dans le champ "nom_compteur"
            Cat_Compteur.CAC_CPT_Compteur=Compteur.CPT_Compteur
        FIN
        HLitRecherchePremier(Categorie,CAT_ID,COMBO_Categorie.COL_IDCategorie)
        SI HTrouve(Categorie) ALORS
            Cat_Compteur.CAC_CAT_Nom=Categorie.CAT_Nom
        FIN
        HAjoute(Cat_Compteur)
        //Ajoute l'enregistrement présent en mémoire dans le fichier de données

        pl_Affichelliste(Vl_cat_compteur)

    FIN
FIN
```

Clic sur BTN_Supprimer

```
// si l'utilisateur ne sélectionne pas de catégorie de compteur on a un message d'erreur
SI TA_Cat_Compteur.COL_CAC_ID= 0 ALORS
    Erreur("Une catégorie de compteur doit être sélectionnée.","Abandon")
    RETOUR
SINON
    HLitRecherchePremier(Cat_Compteur,CAC_ID,Vl_cat_compteur)
    SI HTrouve(Cat_Compteur) ALORS
        HSupprime(Cat_Compteur)

    FIN

    pl_Affichelliste(Vl_cat_compteur)

FIN
```

Initialisation de COMBO_Categorie

```
MaSource.P_CAT_Actif = "1"
```

Sélection d'une ligne de TA_Cat_Compteur

```
Vl_cat_compteur   = TA_Cat_Compteur.COL_CAC_ID
Vpj_Indice        = TableSelect(TA_Cat_Compteur)
```

Sélection d'une ligne de TA_Compteur

```
Vl_compteur   = TA_Compteur.COL_CPT_ID
Vpj_Indice    = TableSelect(TA_Compteur)
```

Bouton gauche double-clic (WM_LBUTTONDBLCLK) sur TA_Compteur

```
Vl_compteur  = TA_Compteur.COL_CPT_ID
Vpj_Indice   = TableSelect(TA_Compteur)
SI Vl_action="CHOI" ALORS
    Ferme()
FIN
```

FEN_SUITE

Procédures

Procédure locale pl_Affiche liste

```
PROCÉDURE pl_Affiche liste(Vl_cat_compteur)
// Suppression de toutes les lignes dans le champ "TABLE_Produit"
TableSupprimeTout(TA_Compteur)
// Libère les ressources d'une requête
HLibèreRequête(REQ_Compteur_liste)

REQ_Compteur_liste.P_CP_Actif=Vrai
// Déclare une requête au moteur HFSQL et exécute cette requête
HExécuteRequête(REQ_Compteur_liste)
SI ErreurDétectée() ALORS Erreur() ; RETOUR
// Affichage
// Remplit un champ de type "Table mémoire" avec tous les enregistrements d'un fichier de données,
d'une vue HFSQL ou d'une requête

FichierVersTableMémoire(TA_Compteur,REQ_Compteur_liste)
// deselectionne toutes les lignes du champ table TA_Compteur
TableSelectMoins(TA_Compteur)

//SI Vpj_Indice=0 ALORS
//  RETOUR
//SINON
// Se place dans le tableau sur la valeur
POUR TOUTE LIGNE DE TA_Compteur
    SI TA_Compteur.COL_CPT_ID=Vl_compteur ALORS
        TableSelectPlus(TA_Compteur,TA_Compteur)
    FIN
FIN
// si aucune sélection, se place sur le 1
SI HNbEnr(REQ_Compteur_liste) > 0 ALORS
    SI TableSelect(TA_Compteur) < 1 ALORS
        TableSelectPlus(TA_Compteur,1)
        Vl_compteur=TA_Compteur.COL_CPT_ID
    FIN
FIN

//FIN


//POUR CAT COMPTEUR
// Supprime toutes les lignes dans le champ Table

TableSupprimeTout(TA_Cat_Compteur)
// Libère les ressources d'une requête (suite à l'utilisation des fonctions HExécuteRequête ou
HExécuteRequêteSQL)
```

```
HLibèreRequête(REQ_Cat_Compteur_SocCat_liste)

REQ_Cat_Compteur_SocCat_liste.P_CAC_SOC_ID = COMBO_Societe.COL_SO_ID
REQ_Cat_Compteur_SocCat_liste.P_CAC_CAT_ID = COMBO_Categorie.COL_IDCategorie
HExécuteRequête(REQ_Cat_Compteur_SocCat_liste)
SI ErreurDétectée() ALORS Erreur() ; RETOUR
// // Remplir le champ TA_Cat_Compteur avec le contenu de REQ_Cat_Compteur_SocCat_liste
FichierVersTableMémoire(TA_Cat_Compteur,REQ_Cat_Compteur_SocCat_liste)

SI HNbEnr(REQ_Cat_Compteur_SocCat_liste) > 0 ALORS
    TableSelectPlus(TA_Cat_Compteur,1)
    Vl_cat_compteur=TA_Cat_Compteur.COL_CAC_ID[1]
FIN
```

FEN_Menu_principal

Code

Déclarations globales de FEN_Menu_principal

```
PROCÉDURE MaFenêtre()
```

FEN_Menu_principal

Code des champs

Sélection du menu de _Menu.OPT_Categorie.OPT_Fiche2

```
Ouvre(FEN_Categorie_liste)
```

Sélection du menu de _Menu.OPT_Categorie.OPT_Liste2

```
Ouvre(FEN_Compteur_liste)
```

Sélection du menu de _Menu.OPT_Categorie.OPT_Societe2

```
Ouvre(FEN_Societe_liste, Paramètre 1, Paramètre N)
```

Sélection du menu de _Menu.OPT_Categorie_compteur

```
Ouvre(FEN_SUITE)
```

Sélection du menu de _Menu.OPT_Combo.OPT_Societe1

```
Ouvre(FEN_Combo)
```

Sélection du menu de _Menu.OPT_Fichier_excel

```
Ouvre(FEN_Excel)
```

Sélection du menu de _Menu.OPT_Fichiers.OPT_Manipuler_chaine_de_caractere

```
Ouvre(FEN_CARACTERE)
```

Partie 3

Requête

REQ_Compteur_VI

Code

Code SQL de REQ_Compteur_VI

```
SELECT
    Compteur.CPT_ID AS CPT_ID,
    Compteur.CPT_Compteur AS CPT_Compteur
FROM
    Compteur
WHERE
    Compteur.CPT_ID <> {P_CPT_ID}
    AND  Compteur.CPT_Compteur = {P_CPT_Compteur}
```

REQ_Compteur_liste

Code

Code SQL de REQ_Compteur_liste

```
SELECT
    Compteur.CPT_ID AS CPT_ID,
    Compteur.CPT_Compteur AS CPT_Compteur,
    Compteur.CPT_Actif AS CPT_Actif
FROM
    Compteur
WHERE
    Compteur.CPT_Actif = {P_CPT_Actif}
ORDER BY
    CPT_Compteur ASC
```

REQ_Societe_liste

Code

Code SQL de REQ_Societe_liste

```
SELECT
    Société.SOC_ID AS SO_ID,
    Société.SOC_Nom AS SOC_Nom,
    Société.SOC_Numero AS SOC_Numero,
    Société.SOC_Actif AS SOC_Actif
FROM
    Société
WHERE
    Société.SOC_Actif = {P_SOC_Actif}
ORDER BY
    SOC_Nom ASC
```

REQ_Societe_VI2

Code

Code SQL de REQ_Societe_VI2

```
SELECT
    Société.SOC_ID AS SOC_ID,
    Société.SOC_Nom AS SOC_Nom,
    Société.SOC_Numero AS SOC_Numero,
    Société.SOC_Actif AS SOC_Actif
FROM
    Société
WHERE
    Société.SOC_ID <> {P_SOC_ID}
    AND  Société.SOC_Nom = {P_SOC_Nom}
```

REQ_Categorie_liste

Code

Code SQL de REQ_Categorie_liste

```
SELECT
    Categorie.CAT_ID AS IDCategorie,
    Categorie.CAT_Nom AS Categorie,
    Categorie.CAT_Actif AS Actif
FROM
    Categorie

    WHERE Categorie.CAT_Actif = {P_CAT_Actif}
ORDER BY
    CAT_Nom ASC,
    IDCategorie ASC,
CAT_Actif ASC
```

REQ_Categorie_VI

Code

Code SQL de REQ_Categorie_VI

```
SELECT
    Categorie.CAT_ID AS CAT_ID,
    Categorie.CAT_Nom AS CAT_Nom
FROM
    Categorie
WHERE
    Categorie.CAT_ID <> {P_CAT_ID}
    AND  Categorie.CAT_Nom = {P_CAT_Nom}
```

REQ_Cat_Compteur_SocCat_liste

Code

Code SQL de REQ_Cat_Compteur_SocCat_liste

```
SELECT
    Cat_Compteur.CAC_ID AS CAC_ID,
    Cat_Compteur.CAC_CAT_ID AS CAC_CAT_ID,
    Cat_Compteur.CAC_CPT_ID AS CAC_CPT_ID,
    Cat_Compteur.CAC_CAT_Nom AS CAC_CAT_Nom,
    Cat_Compteur.CAC_CPT_Compteur AS CAC_CPT_Compteur,
    Cat_Compteur.CAC_SOC_ID AS CAC_SOC_ID
FROM
    Cat_Compteur
WHERE
    Cat_Compteur.CAC_CAT_ID = {P_CAC_CAT_ID}
    AND Cat_Compteur.CAC_SOC_ID = {P_CAC_SOC_ID}
ORDER BY
    CAC_CAT_Nom ASC
```

Partie 4

Collection de procédures

COL_ProcéduresGlobales

Code

Procédure globale pg_bloquer

```
PROCÉDURE pg_bloquer()
RENVOYER opAnnuler
```

Procédure globale pg_locked

```
PROCÉDURE pg_locked(vl_blocage)
vl_User, vl_Adresseip, vl_Machine, vl_Application   sont des chaînes
vl_NumEnr                                               est un entier

SI vl_blocage <> "" ALORS
    // il est bloqué
    vl_NumEnr         = ExtraitChaîne(vl_blocage,2,TAB)
    vl_User           = ExtraitChaîne(vl_blocage,3,TAB)
    vl_Machine        = ExtraitChaîne(vl_blocage,4,TAB)
    vl_Adresseip = ExtraitChaîne(vl_blocage,5,TAB)
    vl_Application    = ExtraitChaîne(vl_blocage,6,TAB)
    HRAZClient()
    HClient.Login         = vl_User
    HClient.Machine       = vl_Machine
    HClient.Application   = vl_Application
    //  SI PAS HEnvoieMessageVersClient(vpj_connexion,"De : "+vpj_connexion..Utilisateur + RC +
    //       "Pouvez vous libérer cet enregistrement ! ",50) ALORS
    //       Erreur("Impossible d'envoyer le message", ErreurInfo())
    //  FIN
    Erreur("Enregistrement bloqué par : "+vl_User,vl_Machine+" "+vl_Adresseip)
    //
    RENVOYER Vrai
SINON
    RENVOYER Faux
FIN
```

Partie 5

Table des matières

Projet Provision_D

Partie 1 **Projet** 3
Code 3

Partie 2 **Fenêtre WINDEV** 5
FEN_Compteur_liste 5
Code 5
Code des champs 5
Procédures 6
FEN_Compteur_fiche 8
Code 8
Code des champs 9