



Extrait du Environnement iSeries

<http://xdocs400.com/spip.php?article213>

Prototype Chargement de sous-fichier avec SQL

- Les articles -



Date de mise en ligne : mardi 9 janvier 2007

Description :

Chargement sous-fichier dynamique avec SQL

Environnement iSeries

Voici un exemple de programme utilisant SQL pour le chargement de sous-fichier. Cet exemple est facilement transposable et peut servir de modèle pour vos écrans (avec SFL) interactif. Il fonctionne avec un programme de service (SGSRVSQ) permettant de gérer les clauses WHERE de requête SQL traité dans un article séparé.

[Voir l'article - Programme de service "SGSRVSQ"](#)

Fonctionnalité du programme MODSQL :

- L'utilisateur peut se positionner sur une ou plusieurs colonnes.
- Le programme se termine par l'activation de la touche F3 ou F12 ou par la sélection d'un code dans le sous-fichier (dans ce cas le programme renvoie en paramètre le code sélectionné).

Ce programme fonctionne avec un fichier DB nommé MODSQLDB voici le script SQL permettant de créer ce fichier.

```
CREATE TABLE MODSQLDB (CODE CHAR (10 ) NOT NULL WITH DEFAULT,  
LIBEL CHAR (50 ) NOT NULL WITH DEFAULT, VALEUR DEC (10 , 2) NOT  
NULL WITH DEFAULT, UNIQUE(CODE) )
```

Pour vos applis vous devrez remplacer MODSQLDB par votre fichier.

Code	Libellé	Valeur
CODE_00001	LIBELLE DU CODE N°1	0,01
CODE_00002	LIBELLE DU CODE N°2	0,01
CODE_00003	LIBELLE DU CODE N°3	0,01
CODE_00004	LIBELLE DU CODE N°4	0,01
CODE_00005	LIBELLE DU CODE N°5	0,01
CODE_00006	LIBELLE DU CODE N°6	0,01
CODE_00007	LIBELLE DU CODE N°7	0,01
CODE_00008	LIBELLE DU CODE N°8	0,01
CODE_00009	LIBELLE DU CODE N°9	0,01
CODE_00010	LIBELLE DU CODE N°10	0,01
CODE_00011	LIBELLE DU CODE N°11	0,01
CODE_00012	LIBELLE DU CODE N°12	0,0 +

Guide sélection multiple

Description technique du module modsql.rpgle

- MODSQL.SQLRPGLE est le source de type SQLRPGLE du module principale du programme.
- MODSQL utilise un fichier écran (MODSQLFM.DSPF) et doit être compilé avec le programme de service SGSRVSQ.RPGLE.
- Vous devez donc créer le module (opt 15 de PDM) puis créer le programme en le liant au programme de service

Prototype Chargement de sous-fichier avec SQL

SGSRVSQ.

- Les zones de positionnements de l'écran (XLIBEL,XCODE,XVALEUR) sont stockées dans La DS de recouvrement ZAFFIC.
- Chaque modification de l'utilisateur d'une de ces valeurs relance l'initialisation de l'écran ainsi que la construction de la requête.
- La partie « WHERE » de la requête est créée par l'appel des 2 procédures du programme de service SGSRVSQ et s'appuie sur les zones de positionnements de l'écran.

- ajout d'une quatrième zone avec positionnement sur l'écran :

il suffit de rajouter les zones à l'écran(Zzone et Xzone), puis dans modsql.rpglr ajouter à la DS de positionnement ZAFFIC Xzone , affecter la valeur dans la procédure ALIM (Zzone=Azone). Enfin dans la prodedure WHERE on invoque la procedure du programme de service (SGWHERN si zone numérique).

```
callp SGWHERA(Xzone : 'A.zone':Ptr_parm:'>=');
```

Code source RPG MODSQL.SQLRPGLE

Pensez à changer la directive copy

```
*----- * Code source généré par SGSQ - Serge GOMES * - Créer module (opt 15 PDM) + CRTPGM PGM(BIB/MODSQL)
BNDSRVPGM(SGSRVSQ) *----- * FICHIER écran FMODSQFM CF E WORKSTN SFILE(SFL01:WRAN01)
*----- * Prototype pgm de service SRVSQL /COPY SERGE/PROTOTYPE,SGSRVSQ
*----- * Pointeur de recouvrement des indicateurs DIndPtr S * INZ(%ADDR(*IN)) D DS
BASED(IndPtr) D ROLLUP 2 2 D SFLDSP 4 4 D SFLDSPCTL 5 5 D SFLCLR 6 6 D SFLEND 7 7 D
SFLNXTCHG 8 8 D FIN_SFL 70 70 *-DS pour alimenter le FETCH DDS_FETCH1 DS D ACODE 10A D ALIBEL 50A
D AVALEUR 8S 2 *-Ds de recouvrement des zones écrans de positionnement DZAFFIC DS INZ D XCODE D XLIBEL D XVALEUR *-Variables
de travail D W_MEM_AFF S LIKE(ZAFFIC) D W_RQT_SQL S 32740 varying D WRQT_W S 32740 varying D W_ORDER S 255
INZ(*BLANK) D W_CURS_OPEN S 1 INZ('N') D W_NB_LIGPAG C 15 D W_SV_SQLCOD S 4B 0 INZ D W_NB_LIGSFL S 4 0 INZ
D W_X S 3 0 INZ D W_FIN_PGM S N INZ(*OFF) DPtr_Parm S * INZ(%ADDR(WRQT_W)) *-Paramètres DPCODE S 10A
*Constantes DWK C "" DWBLEU C X'3a' DWROUGE C X'28' *-----
C *ENTRY PLIST C PARM PCODE C EVAL PCODE = *BLANK *----- C
EXSR INISFL1 C DOW NOT W_FIN_PGM C EXSR ECRAN1 C ENDDO C EVAL *INLR = *ON
*----- C INISFL1 BEGSR C EVAL ZINFO = *BLANK C EVAL SFLEND = *ON C
EXSR PREREQ C EVAL WRAN01 = 0 C EVAL W_NB_LIGSFL = 0 C EVAL SFLCLR = *ON C EVAL SFLDSP = *OFF
C EVAL SFLDSPCTL = *OFF C WRITE FORC1 C EVAL SFLDSPCTL = *ON C EVAL SFLCLR = *OFF C
EXSR PREPAR C IF SQLCOD < 0 *ÆTraitement des erreurs C EVAL ZINFO = WROUGE + 'Erreur SQL N° ' + C
%CHAR(SQLCOD) C ELSE C EXSR DECLAR C EXSR OPEN C EXSR FETCH C IF SQLCOD = 100 C
EVAL SFLEND = *ON *-Msg 'Aucun enregistrement ne correspond ...' C EVAL ZINFO= WBLEU + 'Aucun enregistrement' + C 'ne
correspond à votre sélection' C ELSE C EXSR CHASF1 C END C END C EVAL W_MEM_AFF = ZAFFIC C
ENDSR *----- C ECRAN1 BEGSR C EVAL SFLDSPCTL = *ON C WRITE FORC1 C
WRITE FORB1 *- Lecture écran C READ FORC1 70 *-Traitement touche de fonction C SELECT C WHEN
(*INKC OR *INKL) C EVAL W_FIN_PGM = *ON C EXSR CLOSE C WHEN ROLLUP = *ON C EXSR CHASF1 C
WHEN W_MEM_AFF <> ZAFFIC C EXSR INISFL1 C OTHER C EXSR VERIF1 C ENDSL C ENDSR
*----- C VERIF1 BEGSR C EVAL ZINFO = *BLANK C EVAL SFLNXTCHG = *ON C
Z-ADD 1 WRAN01 C READC SFL01 7070 C DOW FIN_SFL=*OFF AND W_FIN_PGM=*OFF C IF ZCDSEL <>
*BLANK *-Enregistrement sélectionné alimentation paramètres C EVAL PCODE = ZCODE C EVAL W_FIN_PGM = *ON C LEAVE C
ENDIF C READC SFL01 7070 C ENDDO C IF W_FIN_PGM = *OFF C EVAL ZINFO = WROUGE
+ 'Vous devez sélectionner' + C 'un enregistrement' C ENDF C ENDSR *----- C
PREPAR BEGSR *-Préparation instruction C/EXEC SQL C+ PREPARE SLT FROM :W_RQT_SQL C/END-EXEC C ENDSR
*----- C DECLAR BEGSR *-Déclaration curseur C/EXEC SQL C+ DECLARE FILTRE CURSOR FOR SLT C/END-EXEC
```

Prototype Chargement de sous-fichier avec SQL

```

C      ENDSR *----- C OPEN      BEGSR *-Ouverture curseur C      IF      W_CURS_OPEN <> 'O'
C/EXEC SQL C+ OPEN FILTRE C/END-EXEC C      END C      EVAL      W_CURS_OPEN = 'O' C      ENDSR
*----- C CLOSE      BEGSR *-Fermeture curseur C      IF      W_CURS_OPEN = 'O' C/EXEC SQL C+ CLOSE
FILTRE C/END-EXEC C      END C      EVAL      W_CURS_OPEN = 'N' C      ENDSR *----- C
CHASF1 BEGSR *-Charge 1 page SFL01 C      EVAL      SFLDSP = *ON C      EVAL      WRAN01 = W_NB_LIGSFL C      EVAL      W_X = 0 C
      DOW      W_X < W_NB_LIGPAG AND C      SQLCOD <> 100 AND SQLCOD >= 0 C      AND W_NB_LIGSFL < *HIVAL C      EVAL      W_X
= W_X + 1 C      EVAL      W_NB_LIGSFL = W_NB_LIGSFL + 1 C      EVAL      WRAN01 = W_NB_LIGSFL C      MOVEBL *BLANK ZCDSEL C
WRITE SFL01 C      EXSR FETCH C      END C      IF      SQLCOD = 100 OR SQLCOD < 0 C      EVAL      SFLEND = *ON C
ELSE C      EVAL      SFLEND = *OFF C      END C      ENDSR *----- C PREREQ BEGSR C
      EXSR CLOSE C      EVAL      W_RQT_SQL = *BLANK C      EVAL      W_RQT_SQL = 'SELECT * FROM MODSQLDB' C      EXSR WHERE
C      EVAL      W_RQT_SQL = %TRIMR(W_RQT_SQL) + C      ' ' + %TRIMR(WRQT_W ) *-Gestion ORDER BY ... C      EVAL      W_ORDER =
*BLANK C      IF      XCODE <> *BLANK C      EVAL      W_ORDER = %TrimR(W_ORDER) + ',CODE' C      ENDIF C      IF      XLIBEL <>
*BLANK C      EVAL      W_ORDER = %TrimR(W_ORDER) + ',LIBEL' C      ENDIF C      IF      XVALEUR <> 0 C      EVAL      W_ORDER =
%TrimR(W_ORDER) + ',VALEUR' C      ENDIF C C      if      W_ORDER = *BLANK C      EVAL      W_ORDER = ' ' C      Else C
EVAL      W_ORDER = %subst(W_ORDER:3 :%size(W_ORDER)-2) C      Eval      W_ORDER = ' ORDER BY ' + %trim(W_ORDER) C      ENDIF C C
EVAL      W_RQT_SQL = %trim(W_RQT_SQL) + W_ORDER C C *-Optimise la requête C      EVAL      W_RQT_SQL = %TRIMR(W_RQT_SQL) + C
' OPTIMIZE FOR ' + %CHAR(W_NB_LIGPAG) + C      ' ROWS' C      EVAL      W_MEM_AFF = ZAFFIC C      ENDSR
*----- C WHERE      BEGSR *-Cette Routine alimente la partie WHERE de la requête avec les zones de p C      EVAL
WRQT_W = *BLANK C      EVAL      WRQT_W = " *-Gestion des zones alphas - -|- Gestion des zones numériques * SGWHEREA(01:02:03:04) -|Euros
SGWHEREA(01:02:03:04) * 01 => Champs de sélection écran -|Euros 01 => Champs de sélection écran * 02 => Champs de BD -|Euros 02 => Champs de BD * 03 =>
Pointeur sur W_WHE -|Euros 03 => * Pointeur sur W_WHE * 04 => Opérateur -|Euros 04 => Opérateur C      callp SGWHEREA(XCODE :'CODE':Ptr_parm :>=)
C      callp SGWHEREA(XLIBEL :'LIBEL':Ptr_parm :>=) *-Gestion des zones numériques C      callp SGWHEREA(XVALEUR :'VALEUR':Ptr_parm :>=) C
ENDSR *----- C FETCH      BEGSR *-Lecture curseur C/EXEC SQL C+ FETCH FILTRE INTO :DS_FETCH1
C/END-EXEC C      IF      SQLCOD <> 100 AND C      SQLCOD >= 0 C      EXSR ALIM C      END C      ENDSR
*----- C ALIM      BEGSR *-Transfert des zones du fetch vers écran *-Sauvegarde de SQLCOD (utile si vous insérez des
requêtes aux lignes à li C      EVAL      W_SV_SQLCOD = SQLCOD C      EVAL      ZCODE = ACODE C      EVAL      ZLIBEL = ALIBEL C
EVAL      ZVALEUR = AVALEUR *-Ici vos insertions pour traitement ligne à ligne *-Restaure SQLCOD C      EVAL      SQLCOD = W_SV_SQLCOD C      ENDSR
*-----

```

Code source DDS écran MODSQFM.DSPF

Prototype Chargement de sous-fichier avec SQL

```

* ?PROTOTYPE FENETRE POUR MODSQL                A                DSPSIZ(24 80 *DS3)                A                CA03
A                CF12                A                R ASSUME                A                ASSUME                A
    PUTOVR                A                24 80'                A                R WINDKEEP                A                KEEP
        A                OVERLAY                A                R SFL01                SFL                A 08                SFLNXTCHG
A                ZCDSEL                1A B 4 1                A                ZCODE R                O 4 3REFFLD(MODSQLDB/CODE *LIBL/MODSQLDB) A                ZLIBEL R                O 4
14REFFLD(MODSQLDB/LIBEL *LIBL/MODSQLD- A                B)                A                ZVALEUR R                O 4 65REFFLD(MODSQLDB/VALEUR
*LIBL/MODSQL- A                DB)                A                EDTCDE(4)                A                R FORC1                SFLCTL(SFL01)
    A                SFLSIZ(0013)                A                SFLPAG(0012)                A                WINDOW(2 2 16 74 *NOMSGLIN
*NORSTCS- A                R)                A N07                ROLLUP(02)                A                OVERLAY                A
04                SFLDSP                A N04                ERASE(SFL01)                A 05                SFLDSPCTL                A 06
    SFLCLR                A 07                SFLEND                A                WDWBORDER((*DSPATR RI) (*CHAR ' - A
    ))                A                WDWTITLE((*TEXT 'Guide') *CENTER) A                WDWTITLE((*TEXT 'F3/F12=Retour') *L- A
    EFT *BOTTOM)                A                WRAN01                4S 0H                SFLRCDNBR                A                3 3'Code                A
    DSPATR(UL)                A                COLOR(WHT)                A                XCODE R                B 2 3REFFLD(MODSQLDB/CODE *LIBL/MODSQLDB) A
XLIBEL R                B 2 14REFFLD(MODSQLDB/LIBEL *LIBL/MODSQLD- A                B)                A                XVALEUR R                B 2
65REFFLD(MODSQLDB/VALEUR *LIBL/MODSQL- A                DB)                A                EDTCDE(4)                A
3 14'Libellé                - A                '                A                COLOR(WHT)                A                DSPATR(UL)
    A                3 65'Valeur                '                A                DSPATR(UL)                A                COLOR(WHT)                A
    1 3'1=Sélectionner                A                COLOR(BLU)                A                R FORB1                A
    WINDOW(FORC1 )                A                OVERLAY                A                ZINFO                73 O 16 1COLOR(RED)

```

Post-scriptum :

Pourquoi utiliser SQL ? Le même type d'écran développé sans l'utilisation de SQL nous imposerait la déclaration de 3 fichiers logiques et donc de trois boucles de chargements du sous-fichier (ou une boucle imbriquant 3 lectures de fichiers...), l'affectation des clés (KLIST) pour la lecture... Si par la suite on doit ajouter une nouvelle zone sur l'écran il faudra déclarer un nouveau logique...