



Extrait du Environnement iSeries

<http://xdocs400.com/spip.php?article388>

Programme de service pour gérer les erreurs SQL

- Les articles -



Date de mise en ligne : mardi 10 juillet 2007

Environnement iSeries

SRVERRSQL est un programme de service permettant de gérer les erreurs pour les requêtes SQL.

Introduction

Pour gérer les erreurs SQL on s'appuie sur sqlcod ou sqlstt, tous les 2 extraits de la DS sqlca.

[Voir article sur SQLCA](#)

Quand une requête SQL est traitée dans votre programme, SQL place un code retour dans les variables SQLCODE et de SQLSTATE. Les codes retour indiquent le succès ou l'échec du fonctionnement de votre requête.

- Si SQL rencontre une erreur SQLCODE est un nombre négatif et SUBSTR (SQLSTATE, 1.2) n'est pas « 00 », « 01 », ou « 02 ».
- Si SQL rencontre un avertissement, SQLCODE est un nombre positif et SUBSTR (SQLSTATE, 1.2) est « 01 » ou « 02 ».
- Si votre requête SQL est traitée sans rencontrer une erreur ou une condition d'avertissement, SQLCODE est zéro et SQLSTATE est « 00000 »

Description du programme de service

SRVERRSQL contient une procédure exportée appelée SQLErreur(). Cette procédure doit être appelée après une instruction SQL, un message statut sera envoyé si une erreur ou un avertissement est rencontré et que l'on est dans un contexte interactif. Dans un contexte batch, un message programme est envoyé (et sera donc visualisable dans la joblog).

Vous devrez transmettre la DS sqlca en paramètre (rappel => il est inutile de déclarer la ds sqlca dans un programme SQLRPGLE, car cette déclaration est implicite).

Par exemple =>

```
When sqlstt='00000' ; // traitement normal ReturnVar=*on ; When sqlstt='02000' ; // Fin de fichier (%eof) ReturnVar=*off ; Other ; // traitement
des erreurs callp SqlErreur(sqlca) ; ReturnVar=*off ; EndSI ;
```

Si SqlErreur traite une instruction SQL en erreur un message d'échappement (fin du traitement) est envoyé. Pour éviter cela il suffit d'invoquer SqlErreur avec comme 2ème paramètre (paramètre facultatif à *ON par défaut) un indicateur à *OFF.

ex=>

```
D Pstop s n inz(*off) callp SqlErreur(sqlca:Pstop) ;
```

Si Pstop est transmis (à *OFF) alors le traitement continue même quand une erreur est détectée.

Programme de service pour gérer les erreurs SQL

[Voir un exemple d'utilisation complet](#)

Code source du programme

➤ Prototype de SRVERRSQL :

```
d SQLErreur PR D pSqlca 136 const D pStop n Options( *NoPass )
```

➤ Code source du programme de service SRVERRSQL :

```
*----- * @ GOMES serge Programme de service Traitement erreurs SQL * * CRTRPGMOD MODULE(BIBSERGE/SRVERRSQL)
SRCFILE(BIBSERGE/QRPGLESRC) *SRCMBR(SRVERRSQL)OPTION(*EVENTF) DBGVIEW(*SOURCE) REPLACE(*YES) * * CRTSRVPGM
SRVPGM(BIBSERGE/SRVERRSQL) *SRCFILE(BIBSERGE/QSRVSR) TEXT('Programme de service erreurs SQL') *ACTGRP(*CALLER)
*----- H NoMain decedit(',') H bnmdir('QC2LE') * dftactgrp(*no) ACTGRP(*CALLER) bnmdir('QC2LE') H COPYRIGHT('Serge
GOMES') /COPY BIBSERGE/PROTOTYPE,SRVERRSQL *----- * QUSRJOB1 - job informations
*----- D QUSRJOB1 PR extpgm('QUSRJOB1') D jobI0100 70 D jobI_bytes 9b 0 const D
jobI_form 10 const D jobI_jobn 26 const D jobI_jobi 16 const D apierror 120 options(*varsize) * Job information D
jobI0100 ds 70 D jobI_bytes 9b 0 inz(61) D jobI_avail 9b 0 D jobI_jobn 26 inz(*) D jobI_jobi 16 inz D jobI_form
10 inz('JOB10100') D jobI_type 1 *----- * QMHSNDPM - send program messages
*----- D QMHSNDPM PR extpgm('QMHSNDPM') D MessageId 7 const D MessageFile 20
const D MessageData 512 const options(*varsize) D MessageDataL 9b 0 const D MessageType 10 const D CallStkEntry 128
const options(*varsize) D CallStkCount 9b 0 const D MessageKey 4 const D ApiError 120 options(*varsize) D* send program message
D sndpgmmsg ds D msgid 7 inz('CPF9898') D msgfile 20 inz('QCPFMSG QSYS ') D msgdataL 9b 0 inz(512) D msgtype
10 inz(*COMP ') D msgmsgq 11 inz(* ') D msgstack 9b 0 inz(1) D msgkey 4 D msgdata s 512
*----- * api error structure *----- D ApiError ds D ApiErrLP 9b 0
inz(%len(Apierror)) D ApiErrLA 9b 0 inz(0) D ApiErrMsg 7 D 1 D ApiErrDta 104
*----- * sql communication area *----- D SQLCA DS D SQLCAID 8
D SQLCABC 9B 0 D SQLCODE 9B 0 D SQLERRML 4B 0 D SQLERRMC 70 D SQLERRP 8 D
SQLERRD 24 D SQLER1 9B 0 Overlay(SQLERRD:1) D SQLER2 9B 0 Overlay(SQLERRD:5) D SQLER3 9B 0
Overlay(SQLERRD:9) D SQLER4 9B 0 Overlay(SQLERRD:13) D SQLER5 9B 0 Overlay(SQLERRD:17) D SQLER6 4a
Overlay(SQLERRD:21) D SQLWARN 11 D SQLSTATE 5 D P_SQLCA 136 Overlay(SQLCA:1) D Sleep pr 10i 0
ExtProc('sleep') D Seconds 10u 0 value D rc s 10i 0 *----- P SQLErreur B EXPORT D
SQLErreur PI D pSqlca 136 const D pStop n Options( *NoPass ) D wStop s n inz(*on) C/Free if %Parms =2; wStop =
= pStop; endif; P_SQLCA = pSqlca; select; // Erreur détectée when SQLCODE < 0; msgtype = *DIAG; // cpf message... If SQLER1 > 0;
msgid = %editw(%dec(SQLER1:7:0):'0 '); %subst(msgid:1:3) = 'CPF'; else; // cpd message... If SQLER2 > 0; msgid =
%editw(%dec(SQLER2:7:0):'0 '); %subst(msgid:1:3) = 'CPD'; else; msgid = %editw(%dec(SQLCODE*-1:7:0):'0 '); %subst(msgid:1:3) = 'SQL';
%subst(msgfile:1:10) = 'QSQLMSG'; EndIf; EndIf; // Avertissement when SQLCODE > 0; msgid = %editw(%dec(SQLCODE:7:0):'0 ');
%subst(msgid:1:3) = 'SQL'; %subst(msgfile:1:10) = 'QSQLMSG'; // Exécution SQL OK other; msgid = 'SQL' + SQLER6; %subst(msgfile:1:10) =
'QSQLMSG'; endsl;

// message text If SQLERml > 0; msgdata = SQLERRmc; msgdataL = SQLERRml; EndIf; // Retrouve les attributs du Job en cours QUSRJOB1
(jobI0100:jobI_bytes:jobI_form : jobI_jobn:jobI_jobi:Apierror); // Job inter-actif ? If jobI_type = 'I'; msgtype = *STATUS; msgmsgq = *EXT; msgstack =
0; QMHSNDPM (msgid:msgfile:msgdata:msgdataL:msgtype : msgmsgq:msgstack:msgkey:apierror); rc = sleep(1); msgtype = *DIAG; msgmsgq = *;
EndIf; // Message d'échappement If SQLCODE < 0 and wStop; QMHSNDPM (msgid:msgfile:msgdata:msgdataL : *ESCAPE' :*CTLBDY':1:msgkey:apierror);
EndIf; /End-Free P SQLErreur E
```

➤ Source du liage (BIB/QSRVSR). Si le fichier source QSRVSR n'existe pas il faut le créer :

```
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('SRVERRSQL_V01') EXPORT SYMBOL('SQLERREUR') ENDPGMEXP
```