IBM i

Scrivere nel joblog

Marco Riva

MK1

www.markonetools.it

IBM CHAMPION 2021-2022 Power area

Power coffee

Ultimo aggiornamento: 10/06/2022

JobLog

Nel power coffee 22/2021 abbiamo visto come utilizzare i servizi SQL di IBM i per leggere il joblog

Oggi analizziamo i metodi per scrivere nel joblog con SQL, RPG e CLP

Codice operativo DSPLY/1

- Il codice operativo **dsply** consente di inviare brevi messaggi
- Molto limitato
- Lunghezza massima testo: 52 car.
- Il messaggio può anche essere `*M` seguito dall'identificativo del messaggio presente nel file `QUSERMSG`
- E' possibile anche richiedere all'utente di digitare una risposta e riceverla direttamente in una variabile del programma RPG
- Nei lavori interattivi per default il messaggio viene inviato alla coda messaggi esterna `*EXT`, ma può anche essere la coda `QSYSOPR` o una coda messaggi specifica

Codice operativo DSPLY/2

```
dcl-s Msg char(52);
dcl-s Domanda char(50);
dcl-s Risposta char(2);

Msg = 'Messaggio a coda *EXT';
dsply Msg '*EXT';

Msg = 'Messaggio a coda QSYSOPR';
dsply Msg 'QSYSOPR';

Domanda = 'Digita un mese in cifre';
clear Risposta;
dsply Domanda ' ' Risposta;
Msg = 'Hai scelto il mese ' +
Risposta;
dsply Msg;

// messaggio da file QUSERMSG
dsply *MMKT0001;
```

Lunghezza max della variabile per il messaggio è **52**
Se si usa anche la variabile per la risposta la somma di entrambe deve essere al max **52**

```
DSPLY  Messaggio a coda *EXT
```

```
DSPLY  Messaggio a coda QSYSOPR
```

```
DSPLY  Digita un mese in cifre
5
DSPLY  Hai scelto il mese 5
```

Il file messaggi deve essere QUSERMSG.
L'identificativo del messaggio deve essere 3 lettere + 4 cifre

```
Nel mezzo del cammin di nostra vita
```

Scrivere nel Joblog con SQL

LPRINTF procedura per scrivere un messaggio *informativo* nel joblog.
Equivalente alla API `Qp0zLprintf`.

```
exec sql
  call systools/lprintf('Nel mezzo del cammin di nostra vita mi +
                         ritrovai per una selva oscura');
```

MESSAGE_ID	MESSAGE_TYPE	SEVERITY	MESSAGE_TIMESTAMP	MESSAGE_FILE	MESSAGE_TEXT	MESSAGE_SECOND_LEVEL_TEXT
[NULL]	INFORMATIONAL	0	2020-07-18 01:36:38.742420	[NULL]	Nel mezzo del cammin di nostra vita mi ritrovai per u	[NULL]
SQL0420	INFORMATIONAL	30	2020-07-18 01:31:22.548934	QSQLMSG	Carattere nell'argomento CAST non valido.	&N Causa . . . : Un carattere nell'argomento
CPF5029	NOTIFY	30	2020-07-18 01:31:22.548822	QCPFMSG	Errore di corrispondenza dati nel membro QSQPTABL	&N Causa. . . . . : Si è verificato un errore di c
CPF5029	SENDER	30	2020-07-18 01:31:22.548812	QCPFMSG	Errore di corrispondenza dati nel membro QSQPTABL	&N Causa. . . . . : Si è verificato un errore di c
CPD5036	DIAGNOSTIC	10	2020-07-18 01:31:22.548779	QCPFMSG	Errore di corrispondenza dati nel membro QSQPTABL	&N Causa. . . . . : Si è verificato un errore di c
SQL0420	INFORMATIONAL	30	2020-07-18 01:31:22.439007	QSQLMSG	Carattere nell'argomento CAST non valido.	&N Causa . . . : Un carattere nell'argomento
CPF5029	NOTIFY	30	2020-07-18 01:31:22.438872	QCPFMSG	Errore di corrispondenza dati nel membro QSQPTABL	&N Causa . . . . . : Si è verificato un errore di c

da 7.3

portabile in release precedenti: cfr. script di Scott Forstie
https://gist.github.com/forstie/314dde2d3e32ef9b7b83495e6d620fff

API Qp0zLprintf()/1

- L'API **Qp0zLprintf** scrive un messaggio *informativo* nel joblog
- terminare ogni stringa del messaggio con il carattere *new line* `\n` (costante `x'25'` in RPG)

da V4R3

```
// prototipo
dcl-pr WriteToJobLog int(10) extproc('Qp0zLprintf');
 *n pointer value options(*string:*nopass);
end-pr;
dcl-c LF const(x'25');

 WriteToJobLog('Codice articolo ' +
     %trim(WSCART) + ' non trovato.' + LF);
```

API Qp0zLprintf()/2

- L'API può venire chiamata anche con più di 1 parametro
- I parametri successivi al primo sono una *lista di argomenti* che vengono sostituiti nelle variabili segnaposto del 1° parametro

in RPG di fatto la *lista di argomenti* possono essere solo variabili **alfanumeriche**

```
// prototipo
dcl-pr WriteToJobLog int(10) extproc('Qp0zLprintf');
 *n pointer value options(*string:*nopass); // messaggio
 *n pointer value options(*string:*nopass); // segnaposto 1
 *n pointer value options(*string:*nopass); // segnaposto 2
end-pr;
dcl-c LF const(x'25');
dcl-s MsgVar varchar(50) dim(2);
 MsgVar(1) = WSCART;
 MsgVar(2) = 'ER';
 WriteToJobLog('Codice articolo %-10s non trovato. Esito operazione %s'
+ LF : MsgVar(1): MsgVar(2));
```

variabile di tipo stringa allineata a sinistra di lunghezza minima 10 car.

variabile di tipo stringa generica

Comando SNDPGMMSG/1

- Il comando **sndpgmmsg** consente di inviare vari tipi di messaggi (informativi, diagnostica, interrogazione, completamento, eccezione)
- Il testo del messaggio può essere una costante, una variabile o un identificativo di un messaggio contenuto in un file messaggi
- Lunghezza massima: 512 car.
- Per inviare un messaggio di eccezione è obbligatorio specificare un codice messaggio presente in un file messaggi. P.es. il `CPF9898` che contiene solo una variabile segnaposto.

Comando SNDPGMMSG/2

```
chgvar      &msg 'Messaggio informativo a coda *EXT'
sndpgmmsg   msg(&msg) topgmq(*ext) msgtype(*info)

chgvar      &msg 'Messaggio informativo a coda *PRV'
sndpgmmsg   msg(&msg) topgmq(*prv) msgtype(*info)

chgvar      &msg 'Messaggio informativo a coda QSYSOPR'
sndpgmmsg   msg(&msg) tomsgq(*sysopr) msgtype(*info)

chgvar      &msg 'Messaggio informativo a HISTORY LOG'
sndpgmmsg   msg(&msg) tomsgq(*hstlog) msgtype(*info)

chgvar      &msg 'Messaggio di eccezione'
sndpgmmsg   msgid(cpf9898) msgf(qcpfmsg) msgdta(&msg) topgmq(*prv) msgtype(*escape)
  /* dopo questa istruzione il programma termina */
```

API QMHSNDPM/1

- L'API **QMHSNDPM** è praticamente equivalente al comando SNDPGMMSG
- consente di inviare vari tipi di messaggi (informativi, diagnostica, interrogazione, completamento, eccezione) alla coda messaggi del programma o alla coda esterna
- esempio messaggio informativo

da V2R1

```
Message = 'Nel mezzo del cammin di nostra vita';
MsgLen = %len(Message);
SndPgmMsg(' ': ' ':
          Message: MsgLen: '*INFO':
          '*': 0:
          MsgKey:
          Err);
```

testo del messaggio

lunghezza testo

livello corrente nello stack

in allegato al pdf i sorgenti con il prototipo: **APIERREX.RPGLE, SNDMSG_H.RPGLE**

API QMHSNDPM/2

- esempio messaggio di eccezione
  N.B. per i tipi *NOTIFY, *ESCAPE o *STATUS è obbligatorio specificare l'id del messaggio e nome file messaggi

```
Message = 'Codice articolo errato';
MsgLen = %len(Message);
SndPgmMsg('CPF9898': 'QCPFMSG   *LIBL      ':
          Message: MsgLen: '*ESCAPE':
          '*': 0:
          MsgKey:
          Err);
```

id messaggio e nome file messaggi

testo da sostituire nelle variabili segnaposto &*n* del messaggio

API QMHSNDPM/3

- esempio messaggio informativo usando ID messaggio con più di una variabile segnaposto

```
dcl-ds MsgData1;
  Articolo char(15);
  Fornitore char(6);
end-ds;
Articolo = 'ABCDEF';
Fornitore = '000456';
MsgLen = %len(MsgData1);
SndPgmMsg('MKT0001': 'MK1MSGF    *LIBL      ':
          MsgData1: MsgLen: '*INFO':
          '*': 0:
          MsgKey:
          Err);
```

la definizione dei sottocampi deve corrispondere esattamente a quella delle variabili segnaposto

```
MKT0001        Nome
MK1MSGF        Nome
  MK1ILE       Nome, *LIBL, *CURLIB
'Codice articolo &1 del fornitore &2 errato'
```

```
Formati campi dati messaggi:
  Tipo dati  . . . . . .  *CHAR
  Lunghezza  . . . . . .  15
  Byte *VARY o posiz. dec.  0

  Tipo dati  . . . . . .  *CHAR
  Lunghezza  . . . . . .  6
  Byte *VARY o posiz. dec.  0
```

SND-MSG/1

- il codice operativo RPG **snd-msg** invia un messaggio *informativo* (*INFO) o di *eccezione* (*ESCAPE)
- il destinatario definito con la funzione **%target** può essere qualsiasi procedura nello stack delle chiamate. P.es.
  - `%target(*self)`: la procedura corrente
  - `%target(*caller)`: il chiamante
- il messaggio comparirà nel joblog
- il messaggio può essere un testo libero definito nel programma RPG oppure tramite la funzione **%msg** si può specificare un ID di un messaggio contenuto in un file messaggi (*MSGF)

da 7.4 TR6
maggio 2022

SND-MSG/2

- invio di un messaggio informativo

```
snd-msg 'Cognome impiegato obbligatorio';
```

- invio di un messaggio di eccezione

```
snd-msg *escape 'Codice impiegato ' +
  WSEMPNO + ' non trovato';
```

codice CPF9898
gravità 40

SND-MSG/3

- Invio di un messaggio informativo definito in un file messaggi

```
snd-msg %msg('MK10001':'MK1MSGF');
```

- Invio di un messaggio di eccezione (contenente variabili segnaposto) definito in un file di messaggi

```
snd-msg *escape %msg('MK10002':'MK1MSGF':WSEMPNO);
```

- Invio di un messaggio informativo con più variabili segnaposto

```
dcl-ds MK10003_VAR qualified;
  cod_impiegato like(EMPNO);
  stipendio like(SALARY);
end-ds;
snd-msg *escape %msg('MK10003':'MK1MSGF':MK10003_VAR);
```

Riferimenti

- E-mail aziendale: mriva@sirio-is.it
- Blog: www.markonetools.it
- E-mail blog: info@markonetools.it
- Linkedin: www.linkedin.com/in/marcoriva-mk1
- Twitter: @MarcoRiva73
- Facebook: https://www.facebook.com/markonetools/
- YouTube: https://www.youtube.com/channel/UCb47YJQJCzU-5x4nnGzDu-w