

Karl Quon (suite)

Famille : Ma merveilleuse épouse Patricia, ma fille, véritable princesse en formation, et mon courageux fiston.

Animaux : Est-ce que ça comprend les bâtons de golf ?

Sports/Passe-temps : Le golf, la course à pied, la rénovation de la maison et encore le golf.

Ma fin de semaine idéale consiste à :

Voler en jet privé jusqu'à Augusta
Avoir une heure de départ au Augusta National
Voler en jet privé jusqu'à Oahu
Souper aux fruits de mer sur le bord de l'océan
Avoir une heure de départ au Kapalua le lendemain matin
(Ok, je peux me réveiller maintenant)

Mets préférés : Sushi et tout ce que me cuisine mon épouse.

Si je pouvais être quelqu'un d'autre (qu'un conseiller SAS), je voudrais être... un golfeur professionnel.

Lorsque je ne suis pas impliqué dans un projet de consultation SAS, j'aime... regarder mes enfants grandir, m'entraîner pour courir les marathons et, bien sûr, jouer au golf !

Une chose que j'ai apprise sur le terrain en faisant mon travail de conseiller SAS qui, à mon avis, serait utile aux autres utilisateurs SAS :

Sécurité et protection par mot de passe. Le fait d'inclure l'identificateur de l'utilisateur et son mot de passe dans la chaîne d'accès à DBMS ouvre une brèche dans le processus de sécurité, en particulier si elle est laissée ouverte dans un programme .sas ou un fichier autoexec.sas.

Une façon d'y remédier consiste à incorporer cette information dans une macro précompilée. Le piège ici, c'est que la macro est alors vulnérable à une inspection, au moment de l'exécution, par différentes options SAS. Pour contourner cela, utilisez une macro comme celle qui apparaît ci-dessous afin de déceler la présence de ces options SAS – mlogic, mprint, symbolgen.

La macro fonctionne de façon à :

- Conserver en mémoire cache la valeur initiale des options SAS mentionnées ci-dessus.
- Désactiver les options SAS (p. ex., nomlogic, nomprint, nosymbol) dans la macro.
- Établir la connexion à la base de données.
- Réactiver les options SAS à leur valeur initiale.

Par exemple, utilisez cette macro de concert avec les énoncés explicites SQL Pass Through.

Étape 1 – Précompiler la macro.

```
%macro connectDB (database=undefined)
  / store des = 'Database Connection Macro';

  *****;
  * Initializations;
  *****;

  %local COMPNT;
  %local MODULE;

  %let COMPNT = UTILITY;
  %let MODULE = CONNECTDB;

  * following SAS options, if set by the user will present a security gap.;
  * As a result, if any option is set, it will be disabled for the duration;
  * of this macro and reenabled at the end of the macro (if applicable).;

  * the MACROGEN option is not in sashelp.voption, disable it in any case.;
  * Note: SAS Tech Support Note SN-001821 only recommends this option be;
  * used in Version 5.;

  options nomacrogen;

  * determine state of MPRINT option;
  select count(*) into :mprint
    from sashelp.voption
    where upcase (optname) = 'MPRINT'
    and upcase (setting) = 'MPRINT';

  %if (&mprint EQ 1) %then %do;
    options nomprint;
  %end;

  * determine state of MLOGIC option;
  select count(*) into :mlogic
    from sashelp.voption
    where upcase (optname) = 'MLOGIC'
    and upcase (setting) = 'MLOGIC';

  %if (&mlogic EQ 1) %then %do;
    options nomlogic;
  %end;

  * determine state of SYMBOLGEN option;
  select count(*) into :symbolgen
    from sashelp.voption
    where upcase (optname) = 'SYMBOLGEN'
    and upcase (setting) = 'SYMBOLGEN';

  %if (&symbolgen EQ 1) %then %do;
    options nosymbolgen;
  %end;

  *****;
  * Program Logic;
  *****;

  %global prod_db;

  %local user;
```

```

%local password;
%local database;
%local schema;

* Locale: Pebble Beach Database;
%if (%lowcase (%trim (&database)) EQ pebble) %then %do;

    %let user      = kquon;      * db2 userid;
    %let password = golfing;    * db2 password;
    %let database  = pebble;    * db2 database;
    %let schema    = beach;     * db2 schema;

    * db connection string;
    %let db2Parm = user=&user using=&password database=&database schema=&schema;

    * establish connection to database;
    connect to DB2 as pebble (&db2Parm);

%end;

* Locale: Augusta National Database;
%else %if (%lowcase (%trim (&database)) EQ augusta) %then %do;

    %let user      = kaquon;     * db2 userid;
    %let password = golfing;    * db2 password;
    %let database  = augusta;    * db2 database;
    %let schema    = national;   * db2 schema;

    * db connection string;
    %let db2Parm = user=&user using=&password database=&database schema=&schema;

    * establish connection to database;
    connect to DB2 as augusta (&db2Parm);

%end;

%else %do;
    %put ERROR: [&COMPNT:&MODULE] Invalid database name.;
    %let sqlrc = 1;
%end;

* restore SAS options to their original values;
%if (&symbolgen EQ 1) %then %do;
    options symbolgen;
%end;

%if (&mlogic EQ 1) %then %do;
    options mlogic;
%end;

%if (&mprint EQ 1) %then %do;
    options mprint;
%end;

%mend connectDB;

```

Étape 2 – Utiliser la macro

```

* try and get the userid and password;
options mlogic;
options mprint;
options symbolgen;

```

```

proc sql;
%connect (database=pebble);
*<your sql statements here>;
disconnect from pebble;
quit;

```

Étape 3 – Inspecter le journal SAS

Sortie du journal SAS (remarque : l'identificateur d'utilisateur et le mot de passe ne sont pas indiqués).

```

MLOGIC(CONNECTDB): Beginning execution.
MLOGIC(CONNECTDB): Parameter DATABASE has value td_aml
MLOGIC(CONNECTDB): %LOCAL COMPNT
MLOGIC(CONNECTDB): %LOCAL MODULE
MLOGIC(CONNECTDB): %LET (variable name is COMPNT)
MLOGIC(CONNECTDB): %LET (variable name is MODULE)
MPRINT(CONNECTDB): options nomacrogen;
MPRINT(CONNECTDB): select count(*) into :mprint from sashelp.voption where upcase
(optname) = 'MPRINT' and upcase (setting) = 'MPRINT';
SYMBOLGEN: Macro variable MPRINT resolves to 1
MLOGIC(CONNECTDB): %IF condition (&mprint EQ 1) is TRUE
MPRINT(CONNECTDB): options nomprint;
SYMBOLGEN: Macro variable MLOGIC resolves to 1
MLOGIC(CONNECTDB): %IF condition (&mlogic EQ 1) is TRUE
SYMBOLGEN: Macro variable SYMBOLGEN resolves to 1
SYMBOLGEN: Macro variable MLOGIC resolves to 1
SYMBOLGEN: Macro variable MPRINT resolves to 1
MLOGIC(CONNECTDB): %IF condition (&mprint EQ 1) is TRUE
MLOGIC(CONNECTDB): Ending execution.
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.55 seconds
      cpu time           0.02 seconds

```

Exercice pour le lecteur : modifiez cette macro pour les bibliothèques DBMS implicites.