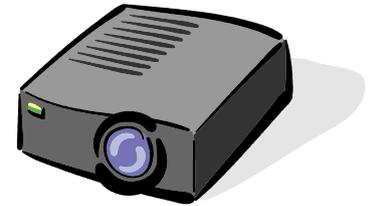


Modernisation et développement d'applications IBM i *Stratégies, technologies et outils*

16 et 17 mai 2011 – IBM Forum de Bois-Colombes



Volubis.fr

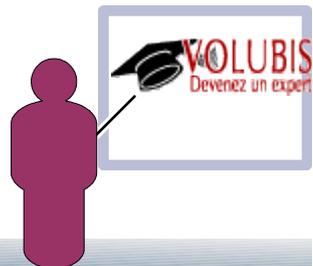
Conseil et formation sur OS/400, I5/OS puis IBM *i*
depuis 1994 !

Christian Massé - cmasse@volubis.fr



Comment réutiliser au mieux le code existant

- ③ Les techniques disponibles :
 - ③ Procédures cataloguées
 - passer un ordre CALL et récupérer un jeu d'enregistrements
 - ③ Créer des fonctions en RPG
 - UDF : User Defined Function
 - ③ Fonctions table (UDTF)
 - la source d'un SELECT SQL est un programme
 - ③ Transformer vos pgms en Services WEB



Comment réutiliser le code existant

Stored Procedure

« Une procédure cataloguée ou procédure stockée est un ensemble d'instruction SQL précompilées et mémorisées dans le dictionnaire » (SQL 2, Christian Marée / Guy Ledant)

Sur DB2, la procédure sera compilée (*en passant par une génération en langage C*), donc il s'agit en fait d'un appel à un exécutable. Compte tenu de ce fait, n'importe quel exécutable peut être considéré comme une procédure cataloguée sur IBM i.

- avec passage de paramètres.

- ces paramètres pouvant être modifiés par le pgm distant.

Syntaxe CALL nom-procédure-----

|
|-(param1,[param2,...])---|



Comment réutiliser le code existant

Stored Procedure

La procédure peut retourner un jeu d'enregistrements
(l'application cliente exécute `CALL` et traite ensuite comme si elle avait passé un `Select`)

Dans la procédure appelée, ajoutez :

```
SET RESULT SETS -----  
!                               !  
!----CURSOR nomcurseur-----!  
!                               !  
!                               !  
!--ARRAY nomdetableau ----FOR x ROWS---
```

S'il s'agit d'un curseur, il doit être ouvert, seules les lignes non lues seront retournées

Pour un tableau, en RPG voyez les DS à occurrences ou à dimensions (DIM)



Comment réutiliser le code existant

Stored Procedure

- Attention, en V5R30, l'ordre SET RESULT SETS possède aussi une clause :

```
                --CLIENT--  
FOR RETURN TO --      --- .  
                --CALLER--
```

qui indique si le curseur ou le tableau (ARRAY) doit être retourné à l'application cliente ou au programme appelant.

- la différence est importante quand on appelle un CL, qui lui même appelle un RPG (par exemple) retournant une DS.



Comment réutiliser le code existant

Stored Procedure

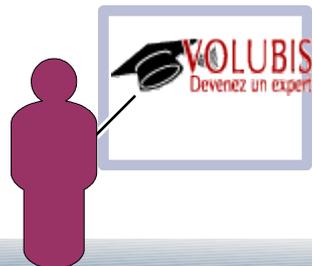
- Vous pouvez enfin préparer tout cela à l'avance en déclarant une fois pour toute la procédure et ses particularités (ce qui est conseillé)

```
CREATE PROCEDURE -nom-proc--(même syntaxe que DECLARE --)---->
```

```
-----  
!                                     !  
!--DYNAMIC RESULT SETS -----n-----
```

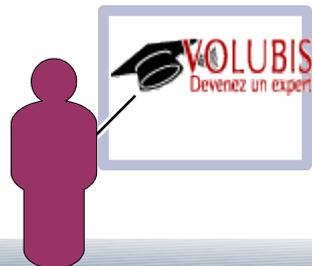
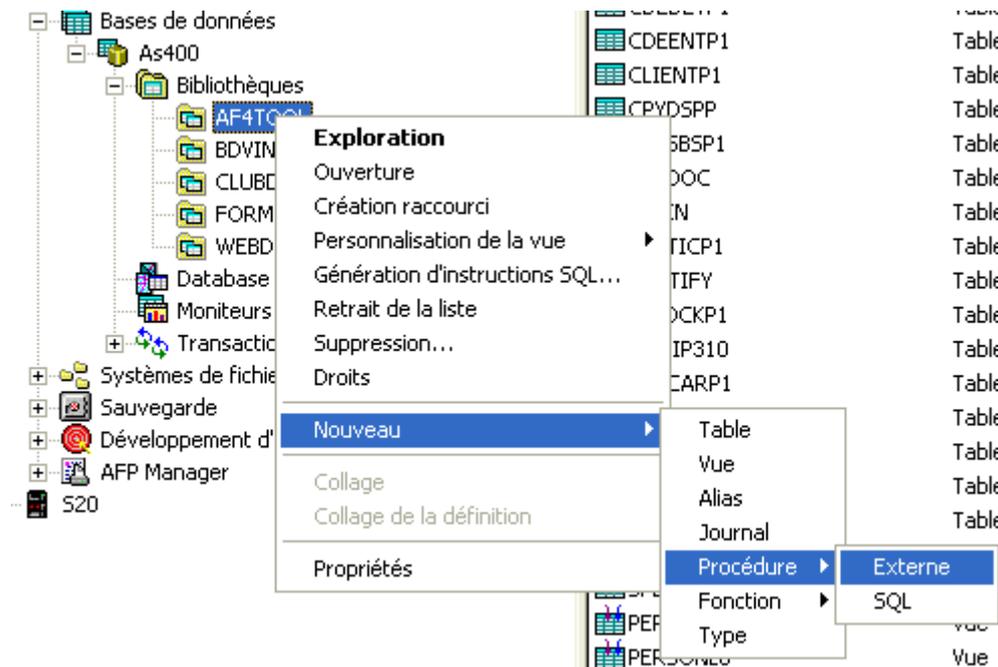
n indique le nombre de "result sets" retournés

- La définition est enregistrée dans les fichiers SYSPROCS et SYSPARMS de QSYS2.



Comment réutiliser le code existant

🕒 Déclaration via System i navigator



Comment réutiliser le code existant

🌀 Déclaration via System i navigator

Nouvelle procédure externe dans AF4TOOL - As400(As400)

Général Paramètres Programme externe

Procédure :

Description :

Nombre maximal de fichiers de résultats :

Même valeur renvoyée à partir d'appels successifs pour des paramètres identiques

Validation des modifications lorsque la commande renvoie à l'appelant

Début d'un nouveau point de sauvegarde lors de l'appel

Accès aux données :

Nom spécifique :

Nouvelle procédure externe dans AF4TOOL - As400(As400)

Général Paramètres Programme externe

Nom du paramètre	Type	Longueur	CCSID	E-S	Relev...	Description
CODART	CHARACTER	5		IN		

Style de paramètre :

SQL

Simple, valeurs indéfinies admises

Simple, valeurs indéfinies non admises

Java

Nouvelle procédure externe dans AF4TOOL - As400(As400)

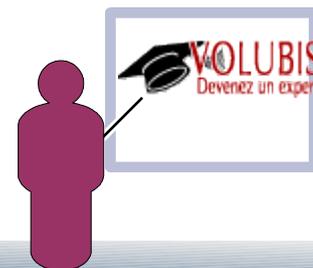
Général Paramètres Programme externe

Programme :

Bibliothèque :

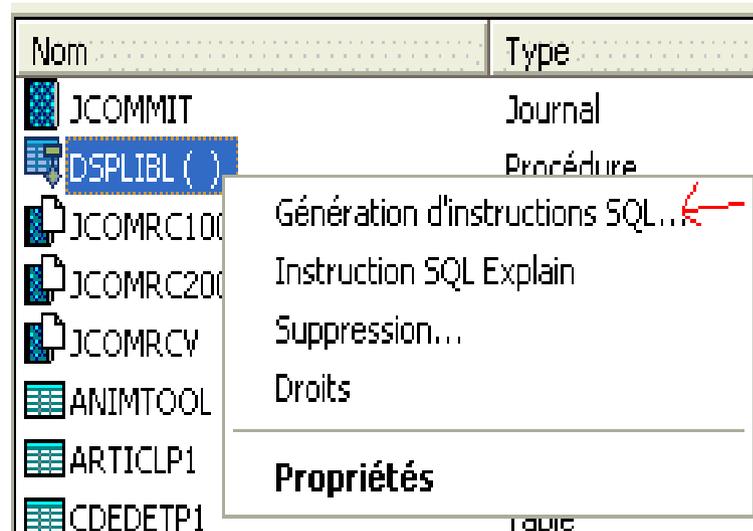
Langage :

Méthode Java :



Comment réutiliser le code existant

- ① Déclaration via System i navigator
- ② Vous pourrez ensuite générer le source de la déclaration (comme tout ce que nous verrons ici) avec :



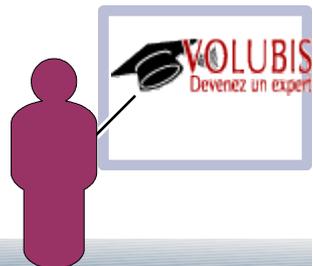
Comment réutiliser le code existant

🌀 Exemple (retourne *LIBL)

1/ enregistrement dans les catalogues

```
CREATE PROCEDURE AF4TOOL/DSPLIBL  
LANGUAGE RPGLE NOT DETERMINISTIC
```

```
CONTAINS SQL  
EXTERNAL NAME AF4TOOL/DSPLIBL  
PARAMETER STYLE GENERAL ;
```



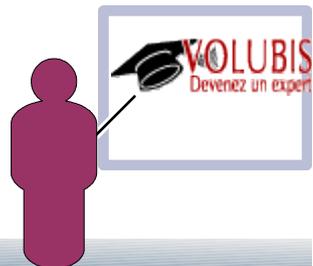
Comment réutiliser le code existant

Exemple (retourne *LIBL)

2/ code du programme (RPGLE)

```
DDSlibl          ds          occurs(265)
D  unebib        10
DUSRlibl        ds
D  tblibl        11          dim(265)
D  i             S           5I 0
D  R             S           5I 0

* prototype pour appel en /free
Ddspliblc       PR          EXTPGM('DSPLIBLC')
D               2750
```



Comment réutiliser le code existant

🌀 Exemple (retourne *LIBL)

```
/free
dspliblc(USRLIBL); // CALL à un CL retournant *LIBL par RTVJOBA
for i=1 to 265;
  if tlibl(i) = *blanks and i > 15; // on est dans usrlibl ==> fin pgm
    leave;
  else;
    if tlibl(i) <> *blanks; // certains noms à blanc dans syslibl
      R = R + 1;
      %occur(DSLIBL) = R;
      unebib = tlibl(i);
    endif;
  endif;
endfor;

EXEC SQL
  SET RESULT SETS  ARRAY :DSLIBL FOR :R ROWS ;

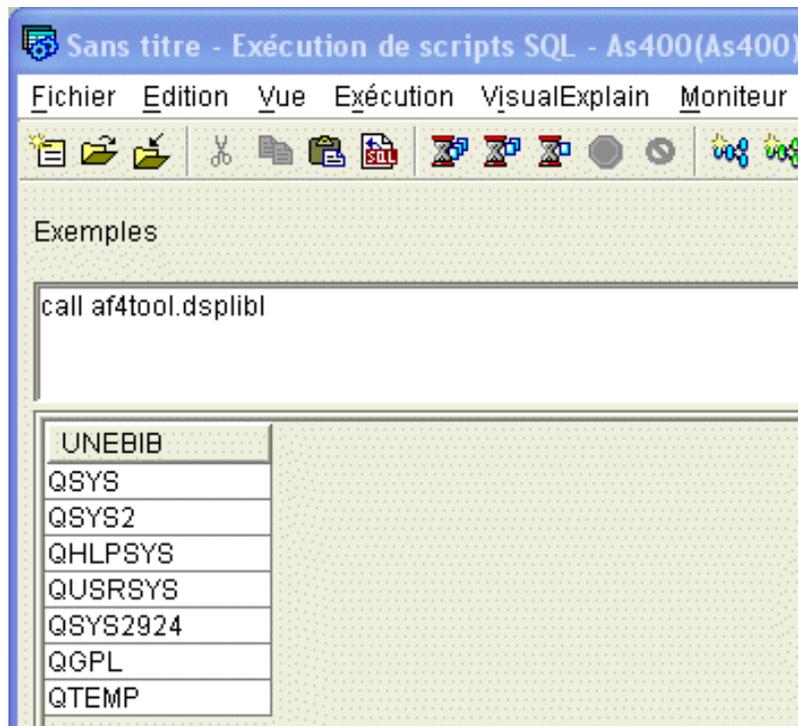
return;

/end-free
```

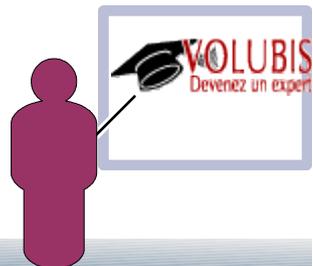


Comment réutiliser le code existant

Résultat



Notez le nom de la colonne, qui est le nom de la zone dans la DS à occurrences



Comment réutiliser le code existant

- La V7 offre la possibilité de récupérer dans un RPG, le « result set » produit par une procédure.

il faut déclarer un RESULT_SET_LOCATOR

```
D  unebib          S          10
D  RS1             S          SQLTYPE(RESULT_SET_LOCATOR)

/free

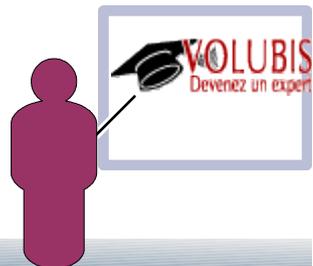
  exec sql
    CALL DSPLIBL;

  if SQLCODE = +466;
    exec sql ASSOCIATE LOCATORS (:RS1) WITH PROCEDURE DSPLIBL;
    exec sql ALLOCATE C1 CURSOR FOR RESULT SET :RS1;
    exec sql fetch c1 into :unebib;

    dow sqlcode = 0;
      // traitement des variables lues...
      exec sql fetch c1 into :unebib;
    ENDDO;

    exec sql close c1;
  ENDIF;

  *inlr = *on;
/end-free
```



Comment réutiliser le code existant

🌀 User Defined Function

- 🌀 une fonction est un programme ou une procédure dans un programme de service enregistré(e) dans les catalogues SQL par CREATE FUNCTION.

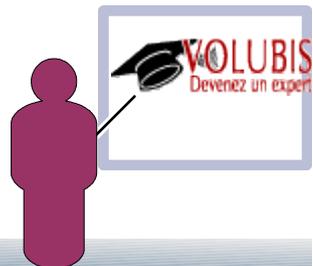
par exemple :

```
CREATE FUNCTION AF4SRCT/NUM2DATE (DEC(8 , 0) ) RETURNS DATE  
  
    EXTERNAL NAME 'AF4SRCT/DT_SQL(DT8_DATE)'      (1)  
    PARAMETER STYLE GENERAL                        (2)  
    RETURNS NULL ON NULL INPUT ;                  (3)
```

(1) fait référence à DT8_DATE dans DT_SQL

(2) le passage de paramètres se fait sans gestion de la val. nulle (uniquement *SRVPGM)

(3) la fonction retourne *null* si un des argument est *null*



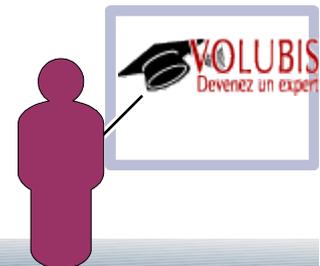
Comment réutiliser le code existant

🌀 User Defined Function

voici le source RPG de la procédure associée à cette fonction :

```
H nomain
* prototype =====
D dt8_DATE      pr          D
D              8 0
* fonction =====
Pdt8_DATE      b          export
d             pi          D
d dt8_recue    8 0
d date        s          D
c      *iso    test(d e)      dt8_recue
c              if          %error
c              return      D'0001-01-01'
c              else
c      *iso    move      dt8_recue      date
c              return      date
c              endif
p              e
```

🌀 Cette même fonction pourrait être utilisée directement en RPG

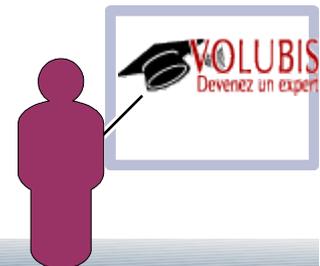


Comment réutiliser le code existant

🕒 User Defined Function

voici le source RPG libre, de la procédure associée à cette fonction :

```
H nomain
* prototype =====
D dt8_DATE      pr          D
D              8 0
* fonction =====
Pdt8_DATE      b          export
d             pi          D
d dt8_recue     8 0
d date         s          D
/free
  test(de) *ISO dt8_recue ;
  if      %error ;
    return %date() ;
  else ;
    date = %date(dt8_recue : *ISO) ;
    return date ;
  endif ;
/end-free
p              e
```



Comment réutiliser le code existant

User Defined Function

la création du programme de service se réalise par :

```
CRTRPGMOD MODULE(DT8_DATE)
```

et

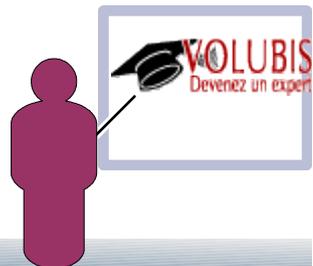
```
CRTSRVPGM SRVPGM(DT_SQL) MODULE(DT8_DATE) EXPORT(*ALL)
```

 cela permet maintenant de passer des ordres SQL comme :

[soit un fichier client avec DATCRT DEC(8 , 0)]

```
SELECT * FROM CLIENTS WHERE NUM2DATE(DATCRT) <> '2011-05-16'
```

```
SELECT * FROM CLIENTS WHERE NUM2DATE(DATCRT) ) = current date
```

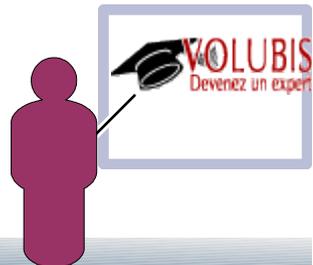


Comment réutiliser le code existant

④ User Defined Function

s'il s'agit d'un programme, ce dernier doit recevoir :

- ④ le(les) paramètre(s) en entrée
- ④ La valeur retour
- ④ le(les) variable(s) indicateur permettant de tester la nullité
- ④ La variable indicateur signalant la nullité de la réponse
- ④ SQLSTATE (5 c.)
- ④ Le nom de la fonction qualifié (517 c.)
- ④ Le nom simple de la fonction (128 c.)
- ④ Message d'erreur retourné par la fonction (1000 c.)



Comment réutiliser le code existant

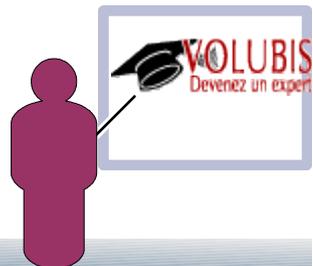
🌀 User Defined Function

Exemple

```
CREATE FUNCTION AF4TEST/DOUBLECAR (CHAR(1) )  
                RETURNS CHAR(2)  
    EXTERNAL NAME 'AF4TEST/DOUBLECAR'  
    PARAMETER STYLE SQL
```

🌀 Notez

- 🌀 Le nom du pgm (sans procédure entre parenthèses)
- 🌀 le style de paramètres SQL



Comment réutiliser le code existant

🕒 User Defined Function

Exemple

```
*=====
* paramètre(s) en entrée et réponse
Dcar          S          1
Dcar2         S          2
* variables indicateur (indication de nullité)
Dcarindic     S          5I 0
Dcar2indic    S          5I 0
* SQLSTATE , '00000' si tout va bien
dSQLSTATE     S          5
* nom de la fonction qualifié
DfonctionQ    S          517      VARYING
* nom (simple) de la fonction
Dfonction     S          128      VARYING
* message d'erreur, s'il y a lieu
Dmessage      S          1000     VARYING
C      *entry      plist
C              parm          car
C              parm          car2
C              parm          carindic
C              parm          car2indic
C              parm          SQLSTATE
C              parm          fonctionQ
C              parm          fonction
C              parm          message
* traitement
C              eval      car2 = car + car
C              return
```



Comment réutiliser le code existant

🕒 User Defined TABLE Function

- 🕒 Depuis la V5R20, SQL admet les fonctions de type TABLE utilisable par

`SELECT * FROM TABLE (nom-fonction()) AS alias`

- 🕒 par exemple (retourne *LIBL):

```
CREATE FUNCTION AF4TEST/TBLIBL ( )  
          RETURNS TABLE (BIB CHAR(10) , LIBTEXT CHAR(50) )  
          EXTERNAL NAME 'AF4TEST/TBLIBL'  
          PARAMETER STYLE DB2SQL                               (1)  
          DISALLOW PARALLEL ;                                   (2)
```

(1) le passage de paramètres se fait avec les paramètres qui suivent.

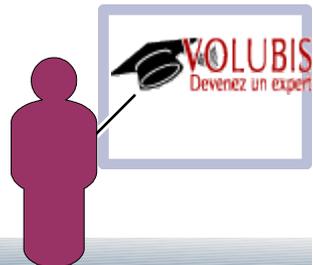
(2) DISALLOW PARRALLEL est obligatoire pour une fonction table



Comment réutiliser le code existant

④ User Defined TABLE Function

- ④ 1/ paramètres reçus (n fois)
- ④ 2/ val. retour
- ④ 3/ indicateurs (integer) pour val nulle de 1.
- ④ 4/ indicateur (integer) pour val. nulle de 2.
- ④ 5/ SQLSTATE
 - '00000' = fonction terminée sans erreur
 - '01Hxx' = warning (xx n'importe pas)
 - '02000' = Fin de fichier : obligatoire !
 - '38Ixx' à '38Zxx' = fonction terminée en erreur ==> SQLCODE négatif
- ④ 6/ Nom qualifié de la fonction
- ④ 7/ Nom simple de la fonction
- ④ 8/ Message de diagnostique



Comment réutiliser le code existant

🌀 User Defined TABLE Function

🌀 OPTIONS

- 9/ scratchpad = zone de dialogue, conservée entre deux appels
UNIQUEMENT AVEC L'OPTION SCRATCHPAD
- 10/ type d'appel
-1= premier , 1 =dernier
0 = autre (appel normal)
-2 = phase d'initiation
2 = phase de terminaison
(-2 et 2 UNIQUEMENT AVEC L'OPTION CALL FINAL)
- 11 / sqludf dans QSYSINC/H)
UNIQUEMENT AVEC L'OPTION DBINFO



Comment réutiliser le code existant



voici le source RPG de la procédure associée à la fonction vue au début :

```
*****
* retourne *LIBL en tant que table
*****
*paramètres
D bib          S          10
D texte       S          50
*indicateurs SQL
D bib_ind     S          5I 0
D texte_ind   S          5I 0
*divers SQL
D SQLSTATE    S           5
D fonction_qual S        139    varying
D fonction_nom S        128    varying
D msg_diag    S          70    varying
D call_type   S          5I 0
* autres variables de travail
D libl        DS
D unposte     S          11    dim(250)
D unebib      S          10    overlay(unposte)
D unblanc     S           1    overlay(unposte:*next)
D i           S         10I 0
D max         S         10I 0

* pgm appelés
DTBLIBL1      PR          2750    EXTPGM('AF4SRCT/TBLIBL1')
D

DTBLIBL2      PR          10      EXTPGM('AF4SRCT/TBLIBL2')
D
D             50
```

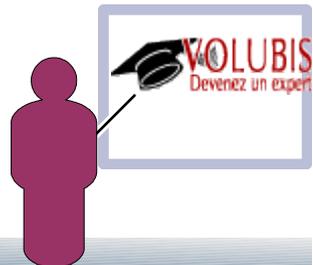


Comment réutiliser le code existant



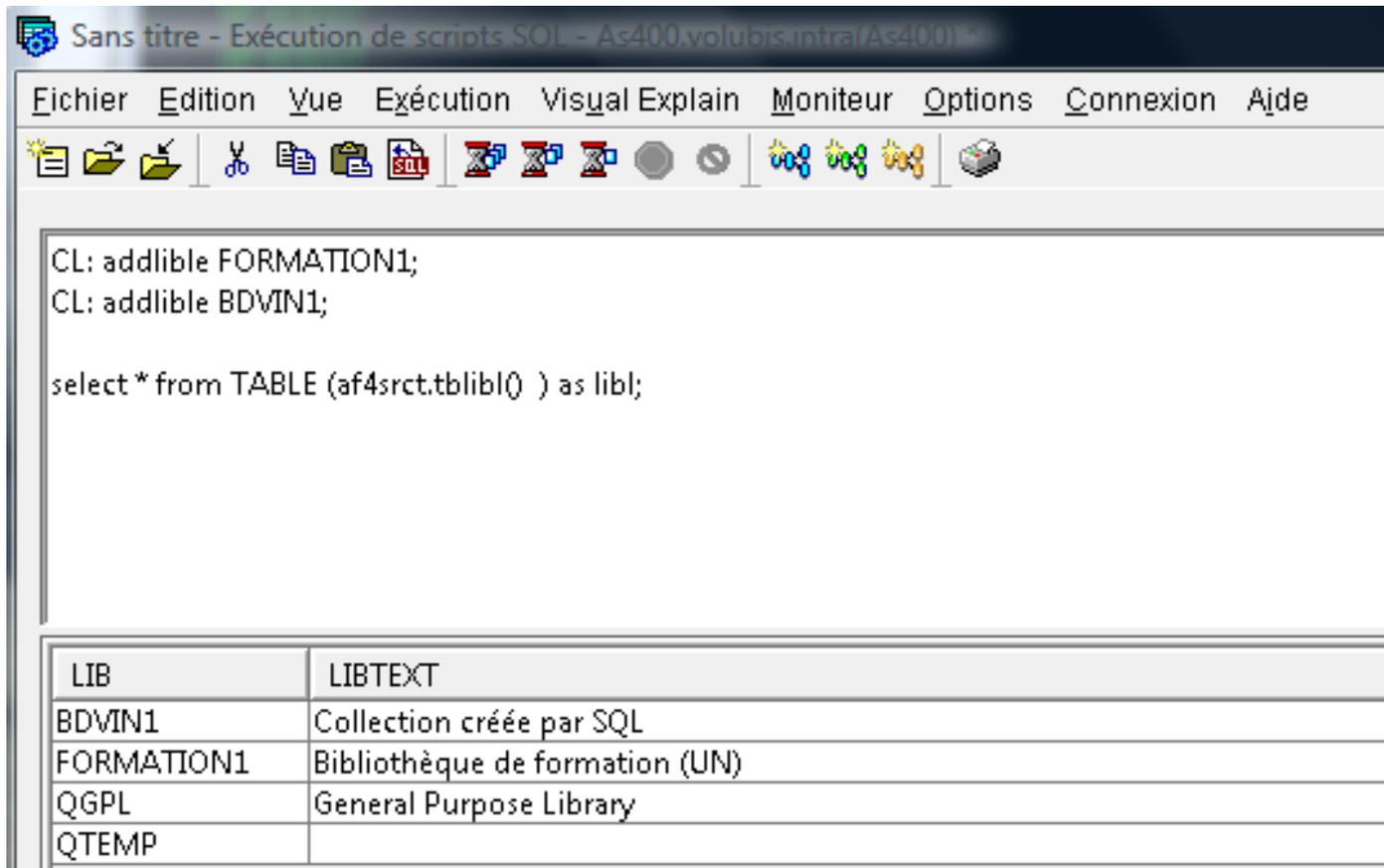
voici le source RPG de la procédure associée à la fonction vue au début :

```
C      *entry      plist
C              parm          bib
C              parm          texte
C              parm          bib_ind>
C              parm          texte_ind
C              parm          sqlstate
C              parm          fonction_qual
C              parm          fonction_nom
C              parm          msg_diag
C              parm          call_type
C      * début du code
C      /free
C          if      call_type < 0 ;
C                  SQLSTATE = '00000' ;
C                  TBLIBL1(lib1) // CL qui retourne *lib1 par RTVJOBA ;
C                  max = %lookup(*blanks : unebib) - 1 ;
C                  if max < 1 ;
C                      max = 250 ;
C                  endif ;
C                  return ;
C          elseif  call_type = 0 ;
C                  I = I + 1 ;
C                  if      I > max ;
C                      SQLSTATE = '02000' ;
C                  else ;
C                      bib = unebib(i) ;
C                      TBLIBL2(bib : texte ) //CL qui retourne UN texte ;
C                  endif ;
C                  return ;
C          else ;
C                  *inlr = *on ;
C          endif ;
C      /end-free
```



Comment réutiliser le code existant

et voici le résultat



Sans titre - Exécution de scripts SQL - As400.volubis.intra(As400)

Fichier Edition Vue Exécution Visual Explain Moniteur Options Connexion Aide

```
CL: addlible FORMATION1;  
CL: addlible BDVIN1;  
  
select * from TABLE (af4srct.tblib1() ) as lib1;
```

LIB	LIBTEXT
BDVIN1	Collection créée par SQL
FORMATION1	Bibliothèque de formation (UN)
QGPL	General Purpose Library
QTEMP	

Nous aurions pu demander :

*Select * from TABLE(af4srct.tblib1()) as lib1 WHERE lib NOT LIKE 'Q%'*



Comment réutiliser le code existant

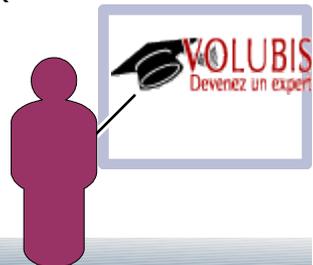


Dans ce cadre, IBM fournit (via PTF SI39822 en V6), la fonction Display_Journal

```
select * From TABLE (Display_Journal(  
    -- bib et journal  
    'BDVIN1', 'QSQJRN',  
    -- bib et récepteur  
    , , ,  
    -- timestamp de début ou null  
    now() - 7 days ,  
    -- séquence de début ou null  
    CAST(null as DECIMAL(21 , 0)),  
    -- code journal  
    , ,  
    -- type d'entrée  
    , , ,  
    -- bib, objet, type, membre , ' ',  
    -- profil utilisateur  
    'QPGMR',  
    -- job  
    , ,  
    -- pgm  
    , , )  
    ) AS jrn
```



permet de lire facilement le contenu d'un ou plusieurs récepteurs de journaux



Comment réutiliser le code existant



Dans ce cadre, IBM fournit (via PTF SI39822 en V6), la fonction Display_journal

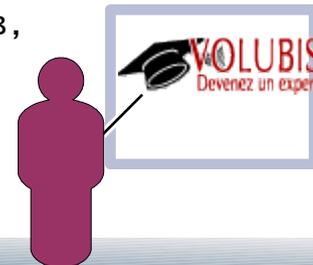
Sans titre - Exécution de scripts SQL - As400.volubis.intra(As400) *

Fichier Edition Vue Exécution Visual Explain Moniteur Options Connexion Aide

```
Select * From TABLE (Display_Journal(  
-- bib et journal  
'BDVIN1', 'QSQRN',  
-- bib et récepteur  
' ' ' ;  
-- timestamp de début ou null
```

ENTRY_TIMESTAMP	SEQUENCE_NUMBER	JOURNAL_CODE	JOURNAL_ENTRY_TYPE	COUNT_OR_RRN	ENTRY_DATA
2011-03-22 16:11:52.413040	209413	F	CB	0	D7D9D6C4E4C3
2011-03-23 02:38:35.364784	209414	D	DW	0	E3C1D7E3C1D7F
2011-03-23 02:38:35.394416	209415	D	DW	0	E3C1D7E3C1D7F
2011-03-23 02:38:35.394416	209416	D	DW	0	E3C1D7E3C1D7F
2011-03-23 02:38:35.394416	209417	D	DW	0	E3C1D7E3C1D7F
2011-03-23 02:38:35.394416	209418	D	DW	0	E3C1D7E3C1D7F
2011-03-23 02:38:35.394416	209419	D	DW	0	E3C1D7E3C1D7F
2011-03-23 02:38:35.394416	209420	D	DW	0	E3C1D7E3C1D7F
2011-03-23 02:38:35.394416	209421	D	DW	0	E3C1D7E3C1D7F
2011-03-23 02:38:35.394416	209422	D	DW	0	E3C1D7E3C1D7F

La colonne contenant les données du poste (ENTRY_DATA) est retournée sous forme de BLOB, castez par CAST(ENTRY_DATA AS CHAR(2000)) pour la voir en clair.



Comment réutiliser le code existant

Service WEB

- Enfin, vous pouvez aussi « exposer » vos programmes en tant que service web.

Pour cela, le plus simple est d'utiliser le serveur d'application intégré à la version 6 : LWI.

c'est lui qui fait tourner DB2 Web Query et qui peut être utilisé pour Web Access.

→ sous HTTPAdmin

(<http://votreas400:2001/HTTPAdmin>)

et créez un nouveau serveur de web services



Comment réutiliser le code existant

The screenshot shows a web browser window titled "HTTP Server Administration on AS400 - Mozilla Firefox". The address bar shows "http://as400:2001/HTTPAdmin". The browser's menu bar includes "Fichier", "Édition", "Affichage", "Historique", "Marque-pages", and "Outils". The browser's toolbar includes navigation buttons, a search bar with "Google", and several tabs. The active tab is "HTTP Server Administration on AS400".

The main content area displays the "IBM Web Administration for i5/OS" interface. The navigation menu includes "Setup", "Manage", "Advanced", and "Related Links". The "Manage" section is expanded to show "All Servers", "HTTP Servers", and "Application Servers".

The "Create Web Services Server" wizard is displayed, with the sub-heading "Specify User ID for Server - Step 1 of 8". The text reads: "Welcome to the Create Web Services Server wizard. A Web services server provides a convenient way to externalize existing programs running on i5/OS, such as RPG and COBOL programs, as Web services. Web service clients can then interact with these i5/OS program based services from the Internet or intranet using Web service based industry standard communication protocols such as SOAP. The clients can be implemented using a variety of platforms and programming languages such as C, C++, Java and .NET. This wizard creates everything needed to run Web services."

The text continues: "The server requires an i5/OS user ID to run the server's jobs. It is recommended that a special user ID be specified to run the server's jobs since this user ID will be given authority to all of the server's objects, such as files and directories."

The "Specify user ID for this server:" section has a help icon and three radio button options:

- Use **default** user ID
- Specify an **existing** user ID
- Create a **new** user ID

A note states: "Note: The default server user ID is QWSERVICE."

At the bottom of the wizard, there are three buttons: "Back", "Next", and "Cancel".

The status bar at the bottom of the browser window shows "Terminé".



Comment réutiliser le code existant

- 🕒 Ici un programme W_RECAP de BDVIN0, attendant une zone PR_CODE, et retournant une DS nommée INFOCENTRE, ce pgm a été compilé avec PGMINFO(*PCML : *MODULE)

Create Web Services Server

Deploy New Service: Specify Location of i5/OS Program Object - Step 3 of 8

The i5/OS object to be externalized as a Web service must be an existing ILE program (*PGM) or service program (*SRVPGM) located on the system. Currently, only program objects written using the COBOL or RPG programming languages are supported.

Specify the library and program object for the Web service. ?

- Specify i5/OS library and ILE program object name (Recommended)

You can specify the program object location by entering the name of the library that contains the program object, as well as the name of the program object. This is the fastest and recommended way to locate the program object.

Library name:

ILE Object name:

ILE Object type: *SRVPGM *PGM

- Browse the integrated file system for the i5/OS program object

Comment réutiliser le code existant

- et précisez le sens d'utilisation des paramètres
(automatiquement découverts par l'assistant grave à PGMINFO)

pour infos, voici un extrait du programme en ce qui concerne les paramètres :

```
DPR_code          S          6  0
DINFOCENTRE      E DS          qualified

C      *entry      plist
C      parm
C      parm          pr_code
C      parm          infocentre
```

Create Web Services Server

Deploy New Service: Select Export Procedures to Externalize as a Web Service - Step 5 of 8

Exported procedures are entry points to a program object and are mapped to Web service operations. A procedure is a set of self-contained high-level language statements that performs a particular task and then returns to the caller. A service program contains one or more procedures. A program contains only one procedure.

The table below lists all exported procedures found in the program object that can be externalized through this Web service. Expand the procedure row to change the default settings for the procedure parameters. The Usage parameter attribute affects what data is sent by clients and what is returned by the Web service. For array type parameters, modifying the Count field may improve Web service performance.

Export procedures: ?

Select	Procedure name/Parameter name	Usage	Data type	Count
<input checked="" type="checkbox"/>	▼ W_RECAP			
	▣ PR_CODE	input	packed	
	▣ INFOCENTRE	output	struct	

Select All Deselect All Expand All Collapse All

Back Next Cancel

Comment réutiliser le code existant

- Les derniers *Group PTF* apportent la possibilité de gérer en variable le nombre d'occurrences d'une structure

Export procedures: ?

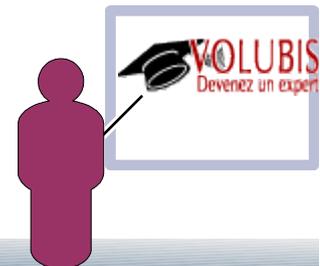
Select	Procedure name/Parameter name	Usage	Data type	Count
<input checked="" type="checkbox"/>	▼ W_RECAP1			
	▣ PR_CODE	input	packed	
	▣ NBPROD	output	int	
	▣ INFOCENTRE	output	struct	500

Select All Deselect All Expand All Collapse All

Back Next Cancel

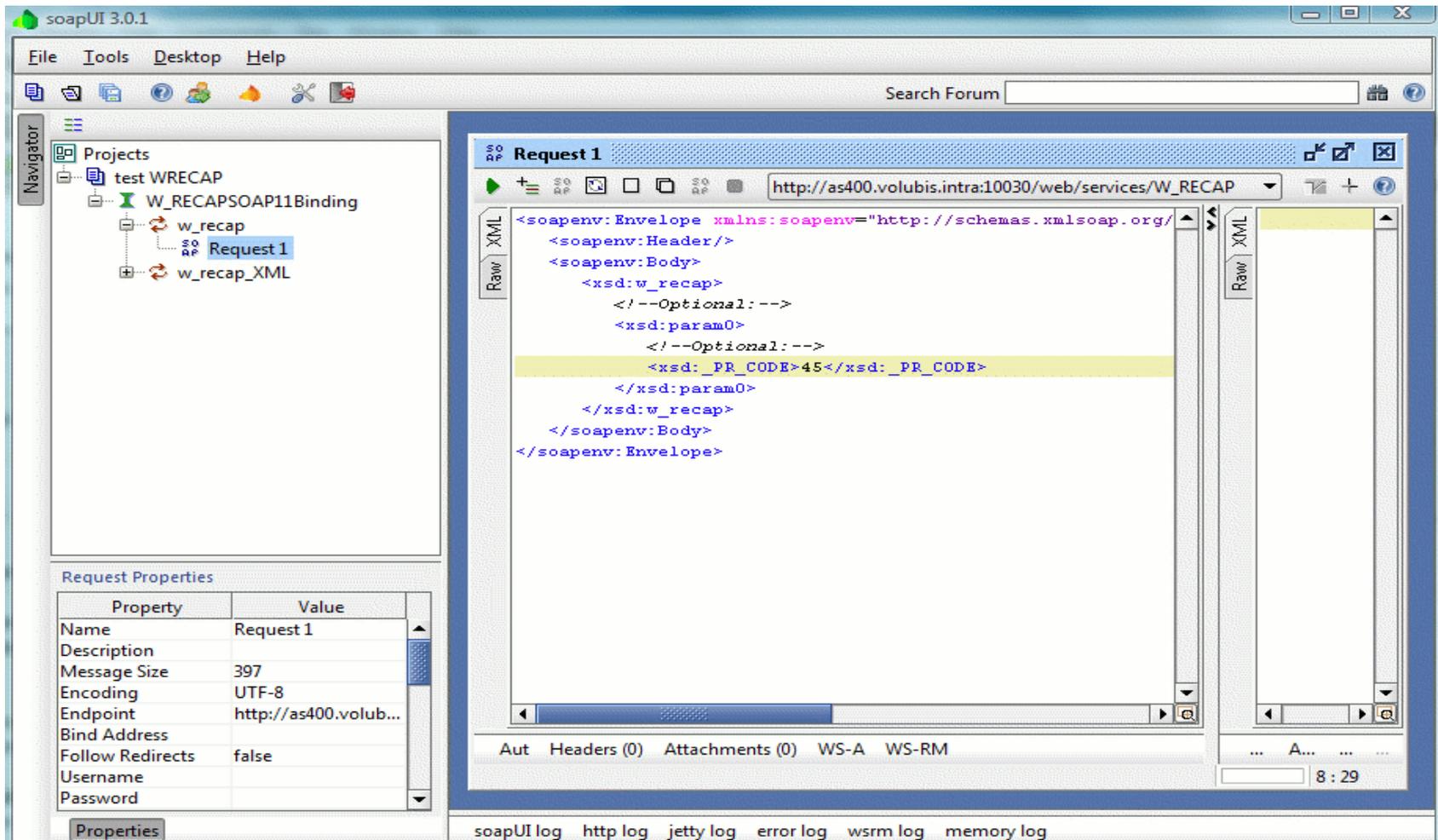
500
NBPROD

- Indiquez ensuite le profil utilisateur de lancement
- la liste de bibliothèques
- Puis confirmez la création sur la dernière page



Comment réutiliser le code existant

- Le déploiement terminé, vous pouvez tester avec l'administration HTTP, mais aussi avec un produit comme *soapUI* (opensource et gratuit):



The screenshot displays the soapUI 3.0.1 application window. The main area shows a SOAP request configuration for 'Request 1' at the endpoint `http://as400.volubis.intra:10030/web/services/W_RECAP`. The XML body is as follows:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap-envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <xsd:w_recap>
      <!--Optional:-->
      <xsd:param0/>
      <!--Optional:-->
      <xsd:_PR_CODE>45</xsd:_PR_CODE>
    </xsd:w_recap>
  </soapenv:Body>
</soapenv:Envelope>
```

The 'Request Properties' table is visible at the bottom left:

Property	Value
Name	Request 1
Description	
Message Size	397
Encoding	UTF-8
Endpoint	http://as400.volub...
Bind Address	
Follow Redirects	false
Username	
Password	

At the bottom of the window, there are log options: soapUI log, http log, jetty log, error log, wsrn log, memory log. A purple silhouette of a person is visible in the bottom right corner, pointing towards the logo.

Comment réutiliser le code existant

- Le déploiement terminé, vous pouvez tester avec l'administration HTTP, mais aussi avec un produit comme *soapUI* (opensource et gratuit):

The screenshot displays the soapUI 3.0.1 application window. On the left, the Navigator pane shows a project named 'test WRECAP' with a sub-project 'W_RECASOAP11Binding' containing a 'w_recap' folder and a 'Request 1' item. Below the Navigator is the 'Request Properties' table:

Property	Value
Name	Request 1
Description	
Message Size	397
Encoding	UTF-8
Endpoint	http://as400.volub...
Bind Address	
Follow Redirects	false
Username	
Password	

The main workspace shows 'Request 1' with a raw XML view of the SOAP envelope. The URL bar indicates the endpoint: `http://as400.volubis.intra:10030/web/services/W_RECAP`. The XML content is as follows:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns:w_recapResponse xmlns:ns="http://w_recap.wsbeans.iseries.com" >
      <ns:return type="iseries.wsbeans.w_recap.W_RECAPRes" >
        <ns:_INFOCENTRE type="iseries.wsbeans.w_recap.INFOCENTRE" >
          <ns:_APPEL00001>Pomerol</ns:_APPEL00001>
          <ns:_CEPAGE>Cabernet Sauvignon</ns:_CEPAGE>
          <ns:_ENCAVE>Non</ns:_ENCAVE>
          <ns:_NBCEPAGE>1</ns:_NBCEPAGE>
          <ns:_NBVIN>1</ns:_NBVIN>
          <ns:_PR_CODE>45</ns:_PR_CODE>
          <ns:_PR_NOM>Château Le Pin</ns:_PR_NOM>
          <ns:_PR_TEL>05 57 51 33 99</ns:_PR_TEL>
        </ns:_INFOCENTRE>
      </ns:return>
    </ns:w_recapResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

At the bottom of the window, the status bar shows 'response time: 243ms (665 bytes)' and a log area with entries for 'soapUI log', 'http log', 'jetty log', 'error log', 'wsrm log', and 'memory log'.