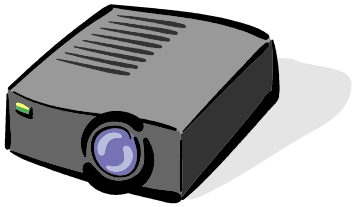
Modernisation, développement d'applications et DB2 sous IBM i
Technologies, outils et nouveautés 2013-2014

13 et 14 mai 2014 – IBM Client Center Paris, Bois-Colombes

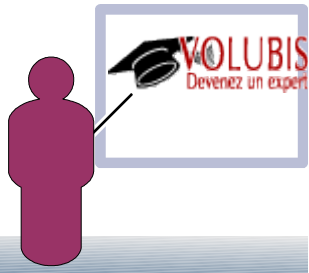


Volubis.fr

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

Dans nos locaux, vos locaux ou par Internet

Christian Massé - cmasse@volubis.fr



Data Structures en RPG

Rappel concernant les Data Structures en RPG

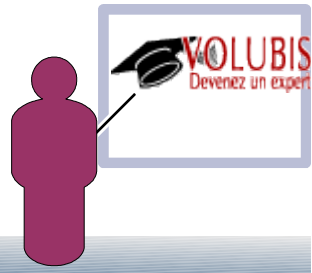
Vous obligez le compilateur à organiser sa mémoire en plaçant certaines informations de manière contiguë

exemples :

```
DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++
DINFOCLI          DS
D nomcli                   1     20
D adrcli                  21     40
D vilcli                  41     55
D cdpst                   56     60   0
```

Ou

```
DINFOCLI          DS
D nomcli                        20
D adrcli                        20
D vilcli                        15
D cdpst                          5   0
```



Data Structures en RPG

Rappel concernant les Data Structures en RPG

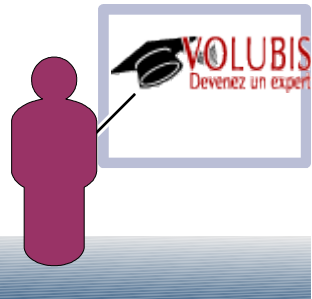
Vous obligez le compilateur à organiser sa mémoire en plaçant certaines informations de manière contiguë. ce qui permet des « découpages »

```
DName++++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++
DINFOCLI          DS
D nomcli                        20
D adrcli                        20
D vilcli                        15
D cdpst                          5   0
D   departementclient...
D                                2   0 OVERLAY(cdpst)
D   bdist                        3   0 OVERLAY(cdpst:*NEXT)
```

. ce qui interdit qu'une même information soit à deux endroits en même temps

```
DINFOCLI          DS
D nom                           20
D adrcli                        20
DINFOFOUR         DS
D nom                           20
D adrfour                       20
```

```
*RNF3315 20 a      000500  Entrée Nom déjà définie dans une spécification de
                           définition. Blancs pris par défaut.
```



Data Structures en RPG

EN V6R10

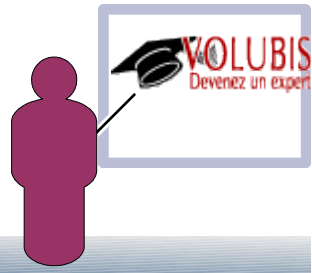
-> la taille maxi des constantes passe à 16380 (la moitié pour UNICODE)

-> la taille maxi des variables caractères passe de 65535c. à 16 Mo !

le mot-clé **LEN** doit être utilisé à la place des colonnes 33-39
pour la définition des variables de grande taille.

EN V7R10 avec PTF SI51094, les déclarations se font en format libre.

```
DCL-DS INFOCLI;
     Nomcli Char(20);
     adrcli Char(20);
     Vilcli Char(15);
     Cdpst  Zoned(5 : 0);
END-DS;
```



Data Structures en RPG

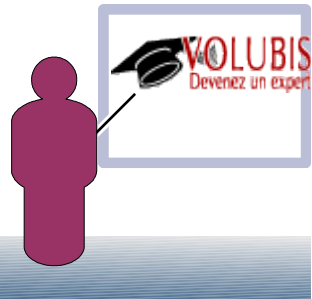
Mots-clés DS :

- EXTNAME(dbf:fmt): nom du fichier(:format) pour une DS externe

Une DS externe est une DS dont les sous zones sont automatiquement générées par copie de la structure d'un fichier qui sert de modèle.

- PREFIX(xxx) : tous les noms de zone sont préfixés

- ALIAS (V7R10) : en cas de CREATE TABLE il est possible d'assigner un nom SQL (long) et un nom système (court) à chaque colonne, les sous zones sont générées à partir des noms longs.



Data Structures en RPG

Mots-clés DS :

- le mot clé LIKEREC permet une définition de DS par copie d'un format d'un fichier **déclaré**, avec choix des zones à copier :

```
LIKEREC(nom-format : *ALL | *INPUT | *OUTPUT | *KEY)
```

*ALL copie de toutes les zones.

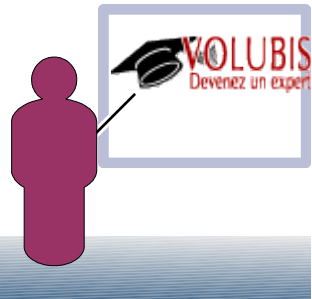
*INPUT les zones en entrée (comme les DS externes)

*OUTPUT les zones en sorties

*KEY uniquement les zones clés.

Une DS peut être utilisée lors d'un CHAIN par :

```
/free
    CHAIN %KDS(nom-de-DS) fichier ;
/end-free
```



Data Structures en RPG

DS évoluées en V5R10 et V5R20

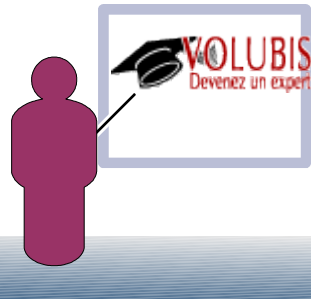
le mot-clé QUALIFIED associé à une définition de DS autorise la manipulation des sous-zones sous la forme :

```
nomDS.nomdezone
```

ce qui permet d'avoir un même nom définit dans deux DS différentes.

ATTENTION, c'est considéré par RPG COMME DEUX CHAMPS DISTINCTS.

```
Dclient           DS                  QUALIFIED
D code                           5
D ville                          3

  client.ville = *blank;
```



Data Structures en RPG

DS évoluées en V5R10 et V5R20

cela permet aussi de définir deux fois la même DS avec LIKEDS

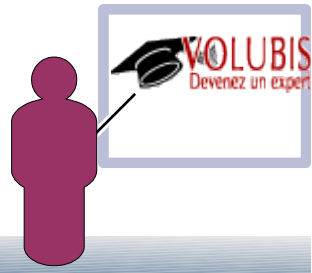
```
Dfournisseur      DS                    LIKEDS(client)
```

remarques

- *les deux DS sont automatiquement qualifiées.*
- *seule la structure de la DS est dupliquée.*
- *les occurrences et les valeurs initiales ne le sont pas.*

  mais vous pouvez préciser sur la deuxième DS : ***INZ(*likeds)***

- *en V6R10, TEMPLATE permet de définir une DS modèle*



Data Structures en RPG

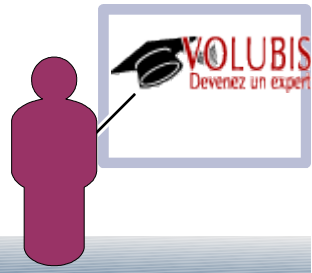
DS évoluées en V5R10 et V5R20

- En V5R20, le mot clé DIM est admis sur les DS (QUALIFIED obligatoire)

```
D article         DS                  DIM(20) QUALIFIED
D  code                          4
D  pointure                      2  0
```

puis manipulation (en format libre) par :

```
/free
  x = x + 1;
  article(x).pointure = '37';
/end-free
```



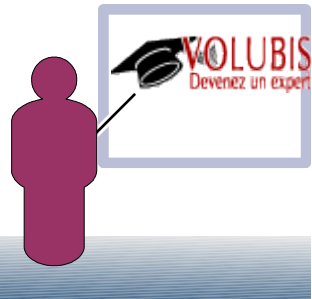
Data Structures en RPG

DS évoluées en V5R10 et V5R20

- et le mot clé LIKEDS est admis sur une sous-zone

```
D dsmodele        DS                  QUALIFIED TEMPLATE
D  matiere                       4
D  couleur                       7

D article         DS                  QUALIFIED
D  codart                        5
D  pointure                      2  0
D  modele                             LIKEDS(DSmodele)
```

```
/free
  article.pointure = 44 ;
  article.modele.matiere = 'Cuir';
  article.modele.couleur = 'Noir';
/end-free
```

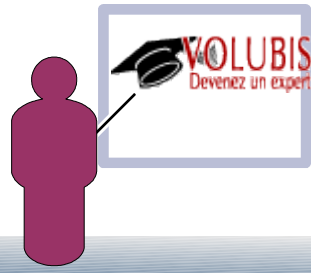


Data Structures en RPG

DS évoluées en V5R10 et V5R20

- Enfin, **Dim** est admis, pour des tableaux à *n* dimensions

```
Dds_dep          DS                    TEMPLATE
D depnom                         25
D ca                             13  2

Dregions         DS                  qualified      dim(50)
D numero                          3  0
D nom                            25
D depart                              likeds(ds_dep) dim(9)


  *
D I              s                   10I 0
D J              s                   10I 0
```



Data Structures en RPG

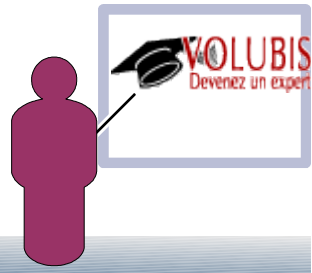
DS évoluées en V5R10 et V5R20

- Enfin, **Dim** est admis, pour des tableaux à *n* dimensions

```
/free
    for i = 1 to %elem(regions) ;
        for j = 1 to %elem(regions.depart) ;

            regions(i).depart(j).ca = 12345;

        endfor;
    endfor;
/end-free
```

EN V7, SORTA et %LOOKUP, utilisent mieux les DS à dimension :

```
sorta(A)  clientDS(*).depart;

pos = %lookup(44 : clientDS(*).depart );
```



Data Structures en RPG

Exemples d'utilisation avec XML-Into

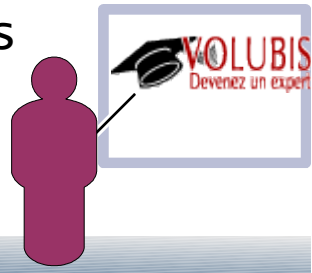
- Ce code opération permet de « parser » du XML

Syntaxe

XML-INTO récepteur %XML(origine : 'options')

Options :

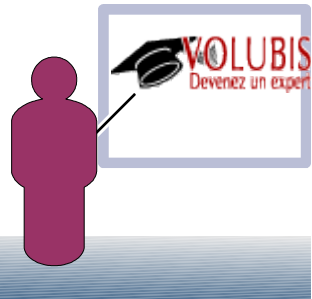
doc=string	*origine* contient du XML
doc=file	*origine* contient un nom de chemin dans l'IFS
Case	sensible ou non à la case
AllowMissing	peut-il y avoir des éléments XML sans équivalent dans la DS
AllowExtra	peut-il y avoir des zones de la DS sans équivalent XML



Data Structures en RPG

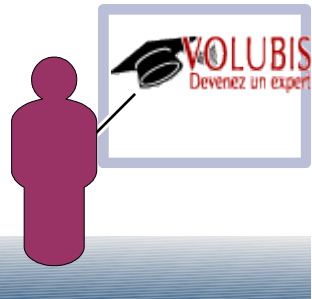
```
 *  test : le nom de DS est le nom de l'élément
 *  il faut que le nom de l'élément soit en minuscule
 *  dans le XML, sauf à indiquer l'option case=upper ou case=any
 *
Ddata             S            1024    inz('+
D                                         <?xml version="1.0" +
D                                               encoding="ISO-8859-1"?>   +
D                                         <cours nom="XML"  module="PGM"> +
D                                         </cours>                        +
D                                        ')
DCOURS            DS
D NOM                              10
D MODULE                           10
```

```
/free
 xml-into cours %xml(data) ; // ou bien %xml(data : 'doc=string')
 *inlr = *on;
/end-free
```



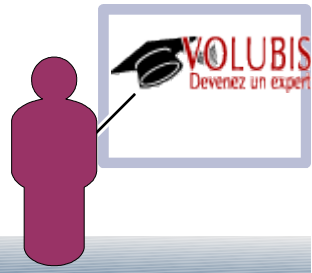
Data Structures en RPG

```
 *
 *  test avec attributs et éléments
 *
 *
Ddata             S             1024    inz('+
D                                          <cours nom="XML"  module="PGM">  +
D                                            <sujet>RPG</sujet>             +
D                                            <texte>manipuler du XML en RPG +
D                                            </texte>                       +
D                                          </cours>                         +
D                                         ')
DCOURS            DS
D NOM                               10
D MODULE                            10
D SUJET                             10
D TEXTE                             50



 /free
  xml-into cours   %xml(data);
  *inlr = *on;
 /end-free
```



Data Structures en RPG

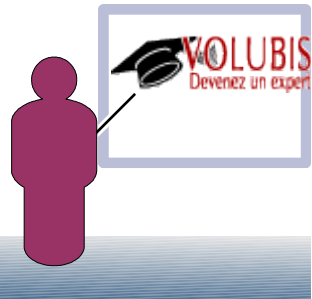
```
 *
 *  test avec une structure imbriquée
 *
 *
Ddata             S             1024    inz('+
D                                          <cours nom="XML"  module="PGM">  +
D                                            <sujet>RPG</sujet>             +
D                                            <texte>manipuler du XML en RPG +
D                                            </texte>                       +
D                                            <motcle>                       +
D                                              <motcl1>RPG</motcl1>         +
D                                              <motcl2>XML</motcl2>         +
D                                              <motcl3>SAX</motcl3>         +
D                                            </motcle>                      +
D                                          </cours>                         +
D                                         ')


DmotclDS          DS
D MOTCL1                        10
D MOTCL2                        10
D MOTCL3                        10
```



Data Structures en RPG

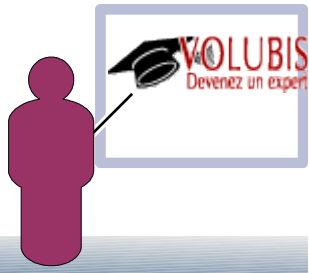
```
DCOURS            DS                  Qualified
D NOM                                 10
D MODULE                              10
D SUJET                               10
D TEXTE                               50
D MOTCLE                              likeds(motclDS)
```

```
 /free
  xml-into cours   %xml(data);
  *inlr = *on;

  //*** en debug ***
  //   COURS.NOM = 'XML        '
  //   COURS.MODULE = 'PGM        '
  //   COURS.SUJET = 'RPG       '
  //   COURS.TEXTE = 'manipuler du XML en RPG                    '
  //   COURS.MOTCLE.MOTCL1 = 'RPG        '
  //   COURS.MOTCLE.MOTCL2 = 'XML        '
  //   COURS.MOTCLE.MOTCL3 = 'SAX        '

 /end-free
```



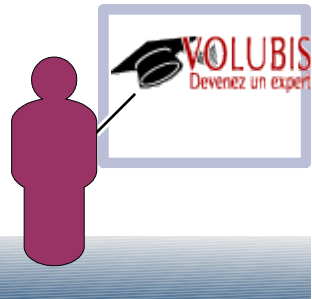
Data Structures en RPG

```
     *
     *   test avec plusieurs occurrences
     *
     *
Ddata             S           2048    inz('+
D                                      <af400 copyright="Volubis">       +
D                                      <cours nom="XML"  module="PGM">   +
D                                        <sujet>RPG</sujet>              +
D                                        <texte>manipuler du XML en RPG  +
D                                        </texte>                        +
D                                        <motcle>                        +
D                                          <motcl1>RPG</motcl1>          +
D                                          <motcl2>XML</motcl2>          +
D                                          <motcl3>SAX</motcl3>          +
D                                        </motcle>                       +
D                                      </cours>                          +
D                                      <cours nom="SQL"  module="PGM">   +
D                                        <sujet>SQL</sujet>              +
D                                        <texte>nouveautés SQL en V5R40  +
D                                        </texte>                        +
D                                        <motcle>                        +
D                                          <motcl1>RECURSIF</motcl1>     +
D                                          <motcl2>RANK</motcl2>         +
D                                          <motcl3>OVER</motcl3>         +
D                                        </motcle>                       +
D                                      </cours>                          +
D                                      <cours nom="I5OS"  module="SYS">  +
D                                        <sujet>V5R4</sujet>             +
D                                        <texte>nouveautés 5.40 de I5/SO +
D                                        </texte>                        +
D                                        <motcle>                        +
D                                          <motcl1>OS</motcl1>           +
D                                          <motcl2>I5</motcl2>           +
D                                          <motcl3>OS4OO</motcl3>        +
D                                        </motcle>                       +
D                                      </cours>                          +
D                                      </af400>                          +
D                                     ')
```



Data Structures en RPG

```
DmotclDS          DS
D MOTCL1                        10
D MOTCL2                        10
D MOTCL3                        10

DCOURDS           DS                    Qualified
D NOM                           10
D MODULE                        10
D SUJET                         10
D TEXTE                         50
D MOTCLE                              likeds(motclDS)

DAF400            DS                     Qualified
D COPYRIGHT                     20
D COURS                               likeds(COURDS) DIM(3)

  /free

   xml-into cours   %xml(data);


   dsply (af400.cours(1).motcle.motcl1); // → RPG

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



Data Structures en RPG

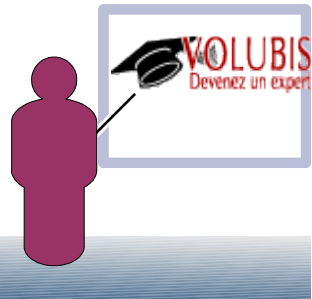
Dans la « vraie » vie, ne pouvant pas prévoir le nombre d'occurrences, nous utiliserons la notion de « *handler* »

```
XML-INTO %handler(proc : zonedecom) %XML(origine : 'options')
```

```
%hanlder fait référence à une procédure (ILE) qui va être appelée
          autant de fois que nécessaire

          Une zone de communication (format libre) va être
             transmise à chaque appel.
```

Prototype de la procédure

```
Dtrtxml         PR            10I 0
D zonedecom                     n
D data                               likeds(ds-reference) dim(50)
D                                    CONST
D nbdata                      10U 0 value
```



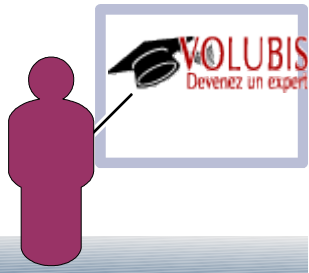
Data Structures en RPG

Dans la « vraie » vie, ne pouvant pas prévoir le nombre d'occurrences, nous utiliserons la notion de « *handler* »

Votre procédure (handler) va recevoir :

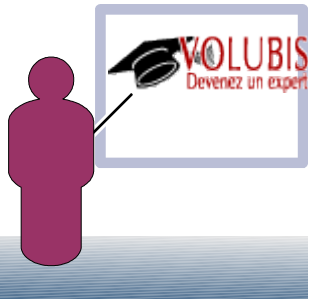
- La zone de communication, dont elle fera ce qu'elle voudra, il s'agit de pouvoir mémoriser et/ou retourner une information.
- Une DS
  - Sans occurrence, la procédure est appelée à chaque occurrence
  - Avec occurrence (DIM(50) par ex.), la procédure reçoit un « paquet » de 50, sauf la dernière fois.
- Une valeur (Integer non signé, sur 10 chiffres), indiquant le nombre d'occurrences réellement transmis,

Enfin votre procédure, qui est en fait une fonction, peut retourner -1 pour arrêter le processus de manière anticipée



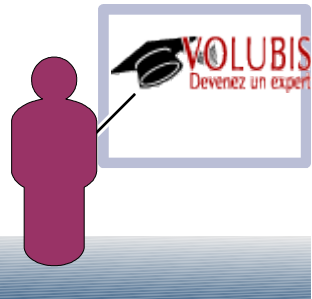
Data Structures en RPG

```
D flagERR         S                 n

  * prototype de la procédure déclarée %handler()
Dtrt10cours       PR             10I 0
D flagERR                          n
D cours10                              likeds(cours)
D                                      dim(10)
D                                      CONST
D nbcours                         10U 0 value

/free
  xml-into       %HANDLER(trt10cours : flagERR)
                 %XML('/af4dir/courshtm/xml/cours.xml'  :
                      'doc=file case=upper path=AF400/COURS') ;
  if flagERR ;
    // traitement en cas d'erreur

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



Data Structures en RPG

```
Ptrt10cours       B
 * paramètres en entrée
D                 PI            10I 0
D flagERR                         n
D cours10                             likeds(cours)
D                                     dim(10)
D                                     CONST
D nbcours                       10U 0 value

 * variables locales
D i               S             10I 0
D plusgrandeDate  S               D   static
 /free
   for i = 1 to nbcours;
   // on cherche la plus grande date
   Monitor;
     if cours10(i).datref > plusgrandedate;
        plusgrandedate = cours10(i).datref;
     endif;
   // en cas d'erreur sur la date
   on-error *all;
     flagERR = *ON;
   endmon;
  endfor;
```

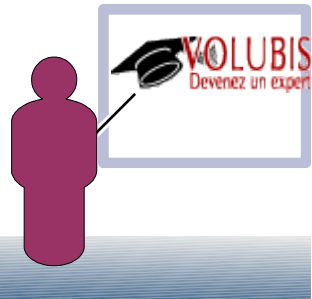


Data Structures en RPG

```
        on-error *all;
       flagERR = *ON;
     endmon;
    endfor;

    if flagERR = *ON;
      return -1;
    else;
      return 0;
    endif;

  /end-free
 Ptrt10cours       E
```

La documentation indique que la procédure doit avoir le prototype suivant

```
 D handler          PR             10I 0
 D commarea                          ?
 D uneDS                                   dim(??)
 D                                         CONST
 D nbelem                           10U 0 value
```



Data Structures en RPG

Dernier point, DS sur les entrées/sorties.

IBM cherche à tuer le « cycle GAP », cela inclus le fait de ne plus utiliser les *spécifs* I et O,

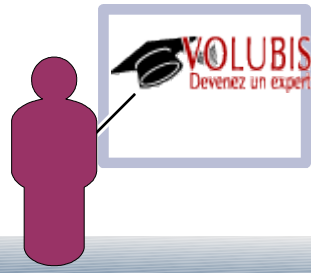
pour cela, TOUS les ordres d'entrée/sortie (EXFMT inclus) acceptent désormais une DS (externe ou non) en dernier argument (avant cela était réservé aux fichiers en interne).

Par exemple

READ fichier **nom-deDS** ;

CHAIN clé fichier/format **nom-de-DS** ;

EXFMT format **nom-deDS** ;



Data Structures en RPG

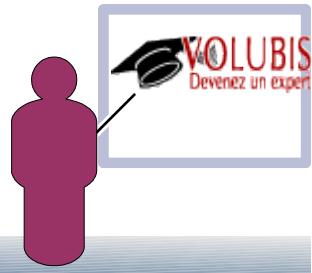
Ainsi le mot-clé **MAIN** en *spécif*. H permet d'indiquer une sous procédure en tant que procédure principale.Cette dernière ne peut être lancée que par CALL.

le source ne contient donc plus que des procédures, la procédure déclarée MAIN doit contenir EXTPGM('le-nom-du-pgm') sur le proptotype

il n'y a plus de cycle GAP , il faut donc fermer explicitement par CLOSE les fichiers (le compilateur signale une erreur RNF7534 de gravité 10)

```
h MAIN(ENTREE)
D ENTREE          PR                  EXTPGM('TESTMAIN') // le nom du source
D                                 5A


P ENTREE          B
D                 PI
D parm1                           5A
 /free

  dsply parm1;

 /end-free
P ENTREE          E
```

nb : Vous ne manipulez plus *INLR



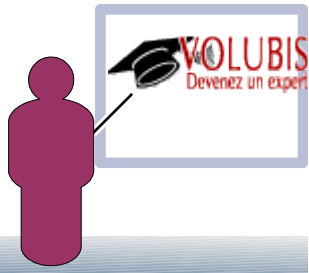
Data Structures en RPG

Et un nom de format peut être qualifié par QUALIFIED en *spécif* F

le nom de format doit être manipulé sous la forme "fichier.format"
(dans notre exemple il s'agit d'une TABLE SQL, d'où le même nom)

il n'a plus à être unique : il n'y a plus de *spécif* I et O pour ce fichier
==> les I/O doivent utiliser une DS :

```
Fvins      IF   E                DISK    QUALIFIED
DinDS              DS                    LIKEREC(vins.vins)


 /free

  read vins INDS;
  dsply  INDS.VIN_NOM;
  Return;

 /end-free
```



Data Structures en RPG

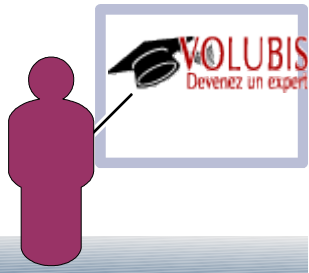
les *spécif*. F sont admises maintenant locales (à l'intérieur des procédures)

les entrées/sorties doivent être réalisées à l'aide de DS.

les fichiers sont fermés automatiquement à la fin de la procédure, sauf à utiliser **STATIC**, qui garde les fichiers ouverts et les variables chargées.

Exemple, utilisation d'une fonction "lecture" qui retourne une donnée Caractère/50 (le nom) à partir d'un code numérique en entrée.

```
H DFTACTGRP(*NO) ALWNULL(*USRCTL)
D lecture          PR                  50
D                                      10I 0 CONST


 /free
  dsply lecture(1) ;
  *inlr = *on;
 /end-free
```

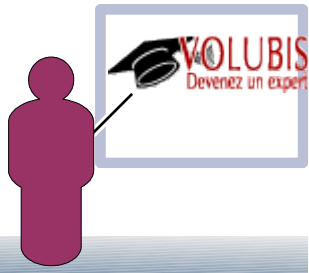


Data Structures en RPG

les *spécif*. F sont admises maintenant locales (à l'intérieur des procédures)

La fonction réalisant la lecture

```
P lecture          B
Fvins      IF   E                DISK    QUALIFIED
D                  PI               50
D code                              10I 0 CONST
DinDS              DS                     LIKEREC(vins.vins)


 /free

  chain code vins INDS;
  return INDS.VIN_NOM;

 /end-free
P lecture          E
```

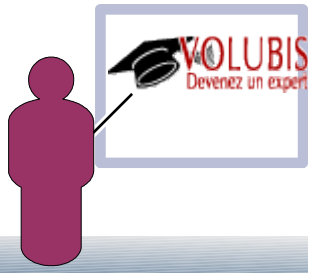


Data Structures en RPG

Dernier point, **LIKEFILE** est autorisé sur un paramètre de procédure, permettant d'utiliser un fichier ouvert en tant que paramètre (transmis ou reçu).

La procédure P1 lit le fichier, passe ce dernier en tant que paramètre à P2 qui fait la mise à jour …

```
* une procédure de mise à jour, qui UPDATE la ligne lue par la procédure
*  principale, le fichier et la DS, sont passés en paramètre.

 * déclarations

H DFTACTGRP(*NO) ALWNULL(*USRCTL)
FVINS      UF   E           K DISK    TEMPLATE QUALIFIED BLOCK(*NO)
FVIN2                                 LIKEFILE(VINS)
F                                      EXTFILE('BDVIN1/VINS')
Dds_modele        DS                  likerec(vins.vins) TEMPLATE
Dmiseajour        PR
D parm1                               LIKEFILE(VINS)
D parm2                               LIKEDS(ds_modele)
DIN               DS                  LIKEDS(ds_modele)
```



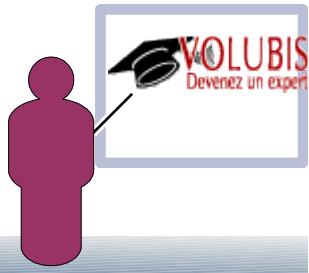
Data Structures en RPG

Corps du programme :

```
  /free
   read vin2 in;
   dow not %eof;
    if in.vin_nom = ' ';
     miseajour(vin2:in) ;
    endif;
    read vin2 in;
   enddo;
   *inlr = *on;
  /end-free

   * cette routine pourrait être externe (*SRVPGM)
 Pmiseajour         B
 D                  PI
 D vin_in                                LIKEFILE(VINS)
 D OUT                                    LIKEDS(DS_MODELE)
  /free
   eval out.vin_nom = 'xxxxxxx' ;
   update vin_in.vins out;
  /end-free
 Pmiseajour         E
```

la procédure reçoit la totalité de l'enregistrement dans OUT

la mise à jour doit aussi avoir lieu par le biais de cette DS (il n'y a plus de *spécif*. I et O)



Data Structures dans un monde n tiers

Les Data structures sont de bons paramètres lors de la réutilisation des programmes RPG existants

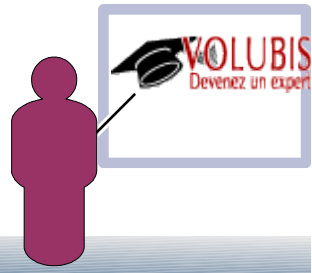
Elles permettent de retourner une série de valeurs

Soit en tant que procédure stockées (RESULT SETS)

Soit en tant que Web services (Paramètre)

Si le serveur d'application est local vous opterez peut-être pour les procédures stockées.

Si le service doit être consommé à distance ou si vous voulez être faiblement couplé avec lui,vous regarderez plutôt les web services.



Data Structures dans un monde n tiers

Procédure stockée

Déclaration

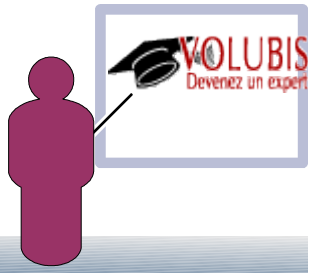
```
DINFOCENTRE     E DS                  qualified dim(50) inz
 *
```

Extrait de code

```
  prod += 1;
  infocentre(prod).pr_code = pr_code;
  infocentre(prod).pr_nom = pr_nom;
  infocentre(prod).pr_tel = pr_tel;
```

Retour de valeur (on peut aussi retourner un curseur SQL ouvert)

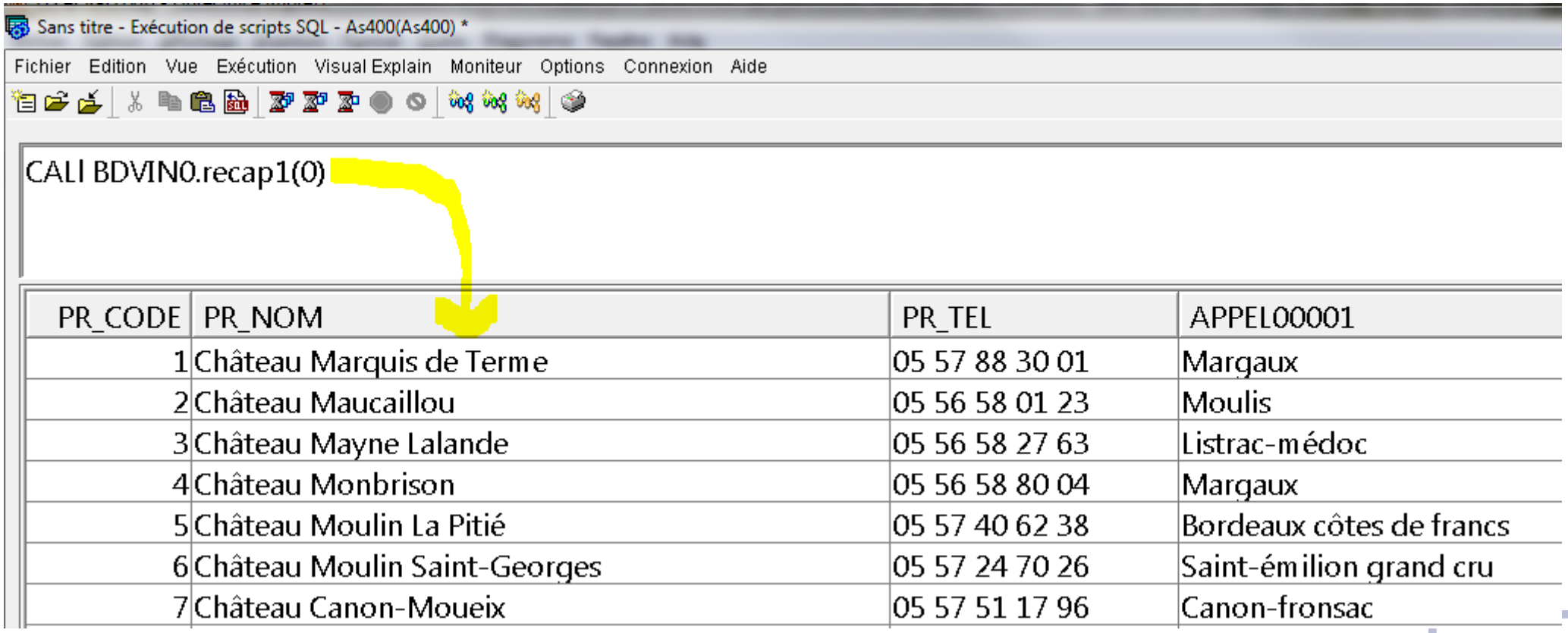
```
 EXEC SQL
   SET RESULT SETS ARRAY :infocentre  FOR :prod ROWS;
 *inlr=*on;
```



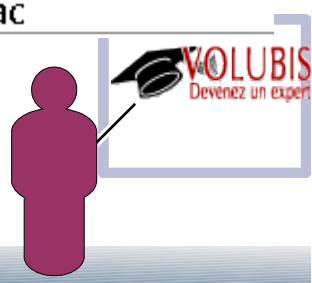
Data Structures dans un monde n tiers

Procédure stockée

Utilisation



PR_CODE	PR_NOM	PR_TEL	APPEL00001
1	Château Marquis de Terme	05 57 88 30 01	Margaux
2	Château Maucaillou	05 56 58 01 23	Moulis
3	Château Mayne Lalande	05 56 58 27 63	Listrac-médoc
4	Château Monbrison	05 56 58 80 04	Margaux
5	Château Moulin La Pitié	05 57 40 62 38	Bordeaux côtes de francs
6	Château Moulin Saint-Georges	05 57 24 70 26	Saint-émilion grand cru
7	Château Canon-Moueix	05 57 51 17 96	Canon-fronsac



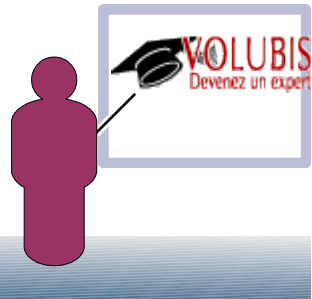
Data Structures dans un monde n tiers

Web service

Déclaration

```
     H ALWNULL(*USRCTL) PGMINFO(*PCML:*MODULE)
     FproducteurIF   E           K DISK
     F                                     extfile('BDVIN0/PRODUCTEUR')
---- 6 lignes retirées. -----

     DINFOCENTRE     E DS                  qualified dim(500)
      *
---- 8 lignes retirées. -----
     Dnbprod           S              10I 0

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

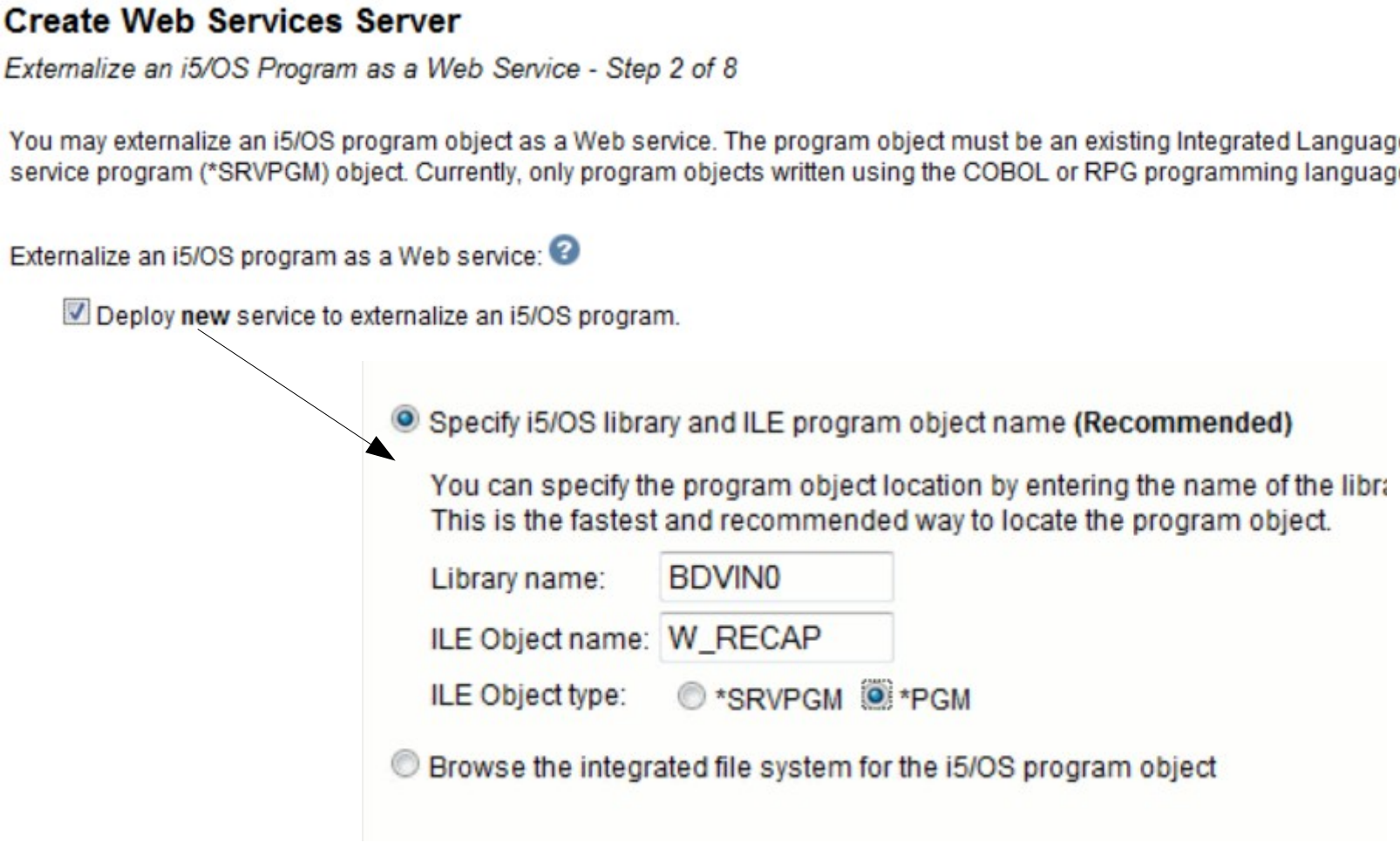
Remarquez PGMINFO en spécif. H !

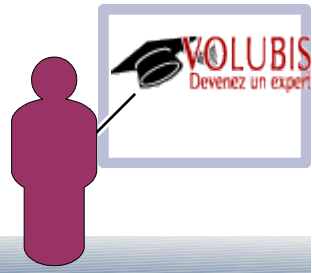


Data Structures dans un monde n tiers

Web service

Déclaration auprès du serveur de Web services

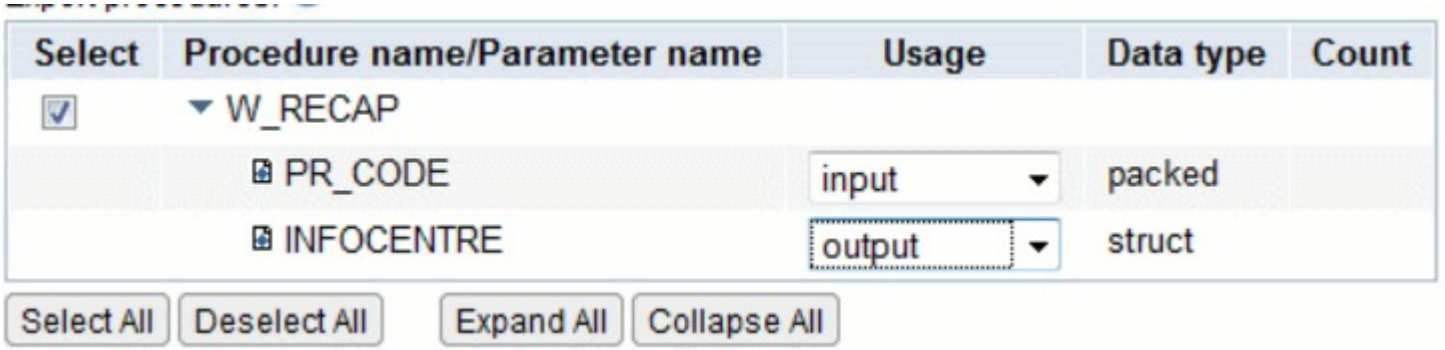




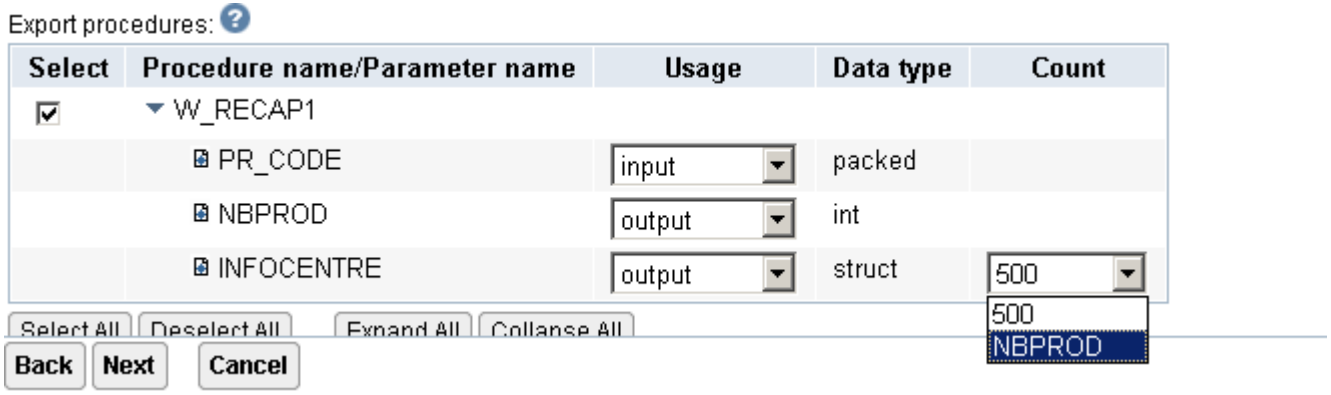
Data Structures dans un monde n tiers

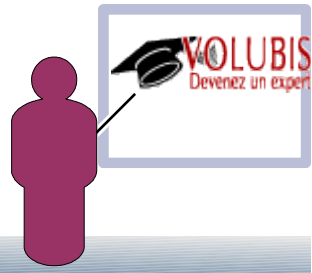
Web service

Déclaration auprès du serveur de Web services



le nombre d'occurrences retournées peut même être une variable (PTF de 2009!)

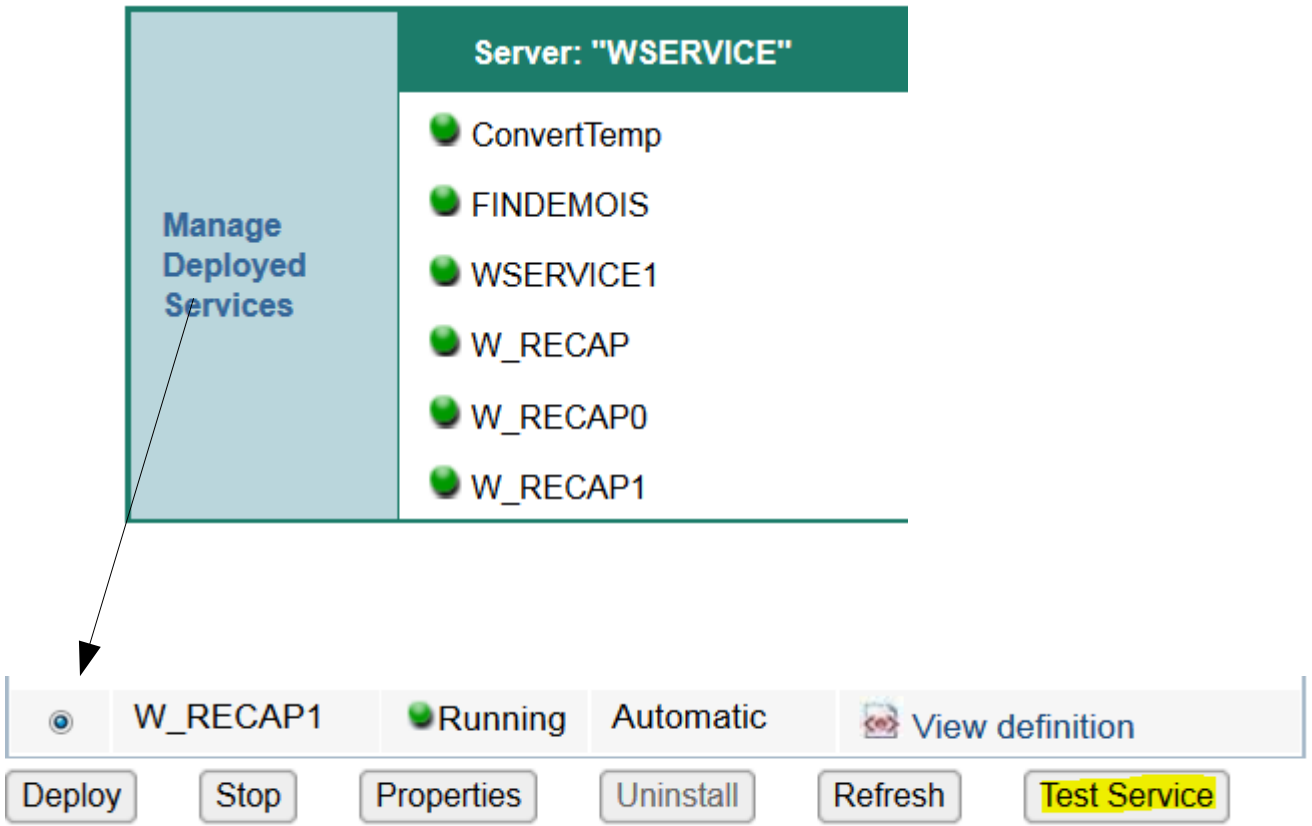


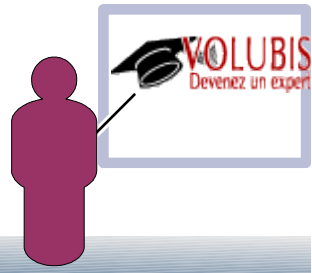


Data Structures dans un monde n tiers

Web service

Test





Data Structures dans un monde n tiers

Web service

Test