

Tutorial 3 Import Export de données via un fichier xml.

Table des matières

- [Objectifs](#)
- [Ressources](#)
- [Pré-Requis](#)

Objectifs

Dans ce TP nous allons utiliser le format `xml` pour échanger des données. C'est une pratique très courante. Cependant l'os400 ne dispose pas d'une commande pour ce faire. Depuis quelques années maintenant IBM se concentre sur SQL. IBM a enrichi SQL sur IBMi avec des instructions pour manipuler des fichiers XML provenant de l'IFS :

- [IFS_READ](#)
- [IFS_WRITE](#)
- [SQL XML programming](#)

Dans notre cas précis nous allons utiliser un script SQL

Ainsi nous allons :

- Écrire un script `xmlOut.sql` dans `acs` pour extraire les films de la table DVD dans un fichier XML.
- Modifier les données depuis un éditeur style `vsc`.
- Écrire un script `xmlIN.sql` pour extraire les données du fichier `csv` avec sélection sur le genre.
- Intégrer ces instructions dans un programme RPG `dspXml.sqlrpgle` pour afficher les films.

en général suite à cette extraction nous écrivons un programme `sqlRPg` qui intègre les modifications dans notre modèle métier.

Ressources

- Environnement
- Temps : 45 mn.

Pré-Requis

- Accéder à internet.

Énoncé

Etape 1 Découvrons le fichier `xml` résultant.

1. Ouvrez le fichier `[listeFilms]`

```
<?xml version="1.0" encoding="UTF-8"?>
<liste_Films>
```

```

    <dvd>
      <code_dvd>10</code_dvd>
      <titre>Star Wars - La menace fantôme</titre>
      <genre>S</genre>
    </dvd>
...
    <dvd>
      <code_dvd>450</code_dvd>
      <titre>Master and Commander</titre>
      <genre>A</genre>
    </dvd>
    <dvd>
      <code_dvd>460</code_dvd>
      <titre>Le dernier Samouraï</titre>
      <genre>A</genre>
    </dvd>
  </liste_Films>

```

c'est un arbre la racine étant `<liste_films>`.

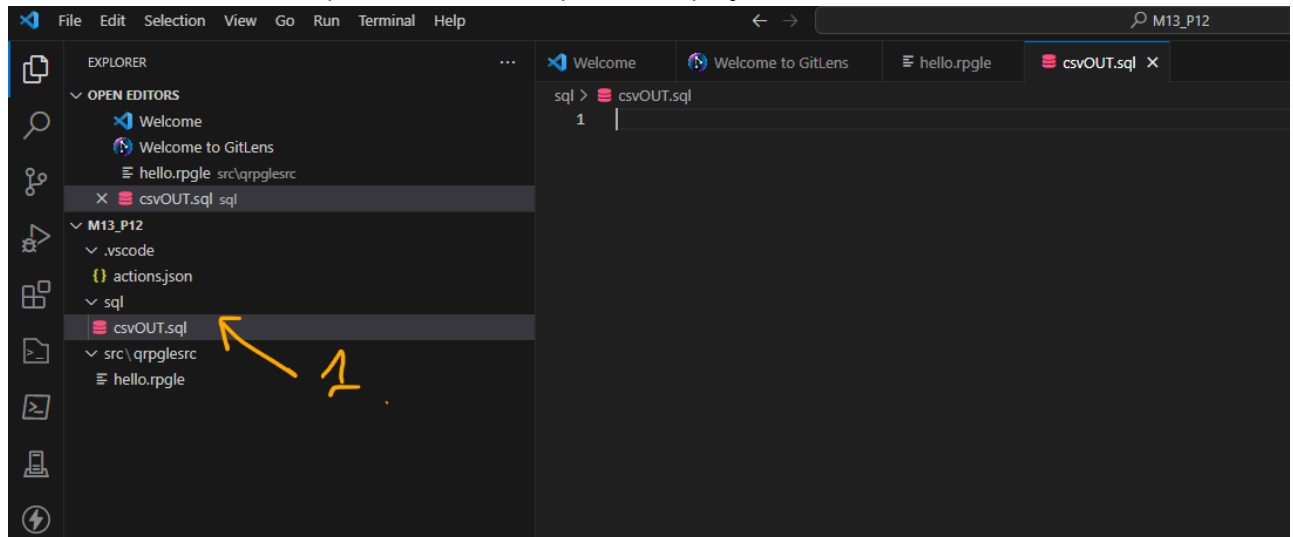
chaque film est une feuille de type `<dvd>`.

un dvd est composé :

- d'un code `<code_dvd>`
- d'un titre `<titre>`
- d'un genre `<genre>`

Etape 2 XML OUT

1. créer un fichier xmlOUT.sql dans le dossier sql de votre projet.



Nous allons utiliser l'instruction `XMLGROUP`.

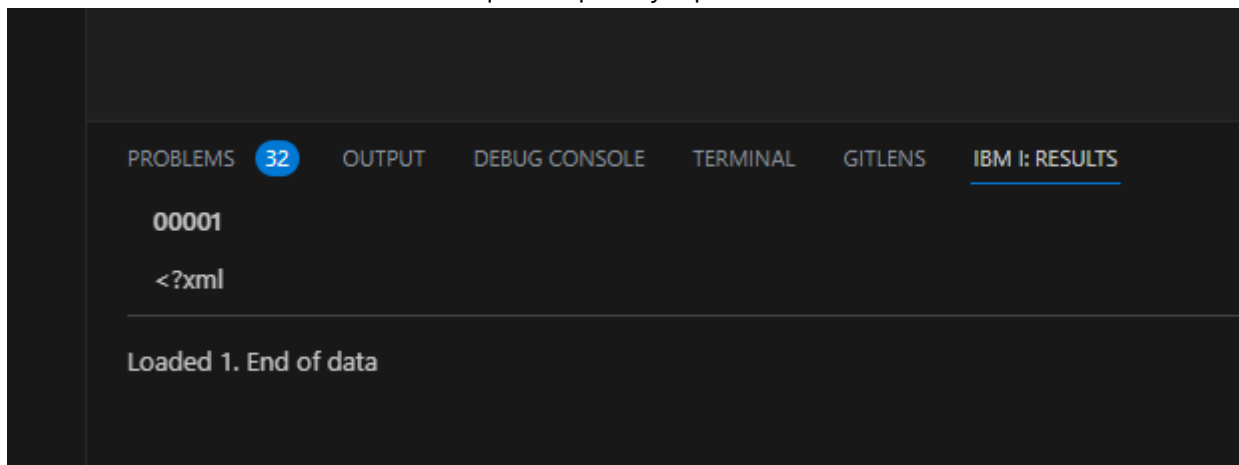
Cette fonction regroupe toutes les lignes sélectionnée pour en former un document xml.

- `TRIM(coddvd) AS "code_dvd"` permet de préciser que le contenu de la zone `coddvd` sera le contenu de la balise xml `<code_dvd>`.
`<code_dvd>460</code_dvd>`
- `OPTION ROW "dvd"` indique que chaque ligne sera une feuille ayant pour nom `<dvd>`
- `ROOT "liste_Films"` indique que la racine `root` se nommera `<liste_Films>`.

1. ajoutons ce ligne dans notre script.

```
-- creation d'un arbre <liste_films> en créant des feuilles <dvd> pour chaque
ligne de notre table DVD.
SELECT XMLGROUP(TRIM(coddvd) AS "code_dvd",
                TRIM(titre) AS "titre",
                genre AS "genre"
                OPTION ROW "dvd" ROOT "liste_Films")
FROM yv.dvd;
```

si vous l'exécutez dans vs code vous remarquerez qu'il n'y a pas de résultat lisible.



En effet cette fonction renvoie un typer xml qui n'est pas interpretable dans vs code.

Néanmoins si vous lancez cette requête dans acs vous pourrez voir le résultat.

```
13
14 SELECT XMLGROUP(TRIM(coddvd) AS "code_dvd",
15                 TRIM(titre) AS "titre",
16                 genre AS "genre"
17
18                 OPTION ROW "dvd" root "liste_films")
19 FROM yv.dvd;
20 SELECT XMLELEMENT (NAME "Liste_Films",
21                  XMLAGG(XMLELEMENT (NAME "item", p.name)))
22 FROM Product p;
23 VALUES
24   (XMLSERIALIZE(SELECT XMLGROUP(TRIM(coddvd) AS "code_dvd",
25                                TRIM(titre) AS "titre",
26                                genre AS "genre"
27                                OPTION ROW "dvd" ROOT "liste_Films")
28 FROM yv.dvd) AS CLOB(10000000) INCLUDING XMLDECLARATION ) ;
```

00001

```
<liste_films><dvd><code_dvd>10</code_dvd><titre>Star Wars - La menace fantôme</titre><genre>S</genre></dvd><dvd><code_dvd>20</code_dvd>
```

Nous allons devoir transformer le résultat xml en format caractère pour qu'il soit lisible mais surtout pour que l'on puisse l'insérer dans un fichier texte sur l'ifs. Pour ce faire nous utilisons l'instruction [XMLSERIALIZE](#).

1. ajouter cette ligne dans votre script.

```
-- transformons le document XML en CLOB big big chaîne de caractère
SELECT xmlserialize(XMLGROUP(TRIM(coddvd) AS "code_dvd",
                              TRIM(titre) AS "titre",
                              genre AS "genre"
```

```
OPTION ROW "dvd" ROOT "liste_Films") as CLOB(10000000) INCLUDING
XMLDECLARATION)
FROM yv.dvd;
```

ici nous transformons le document xml créé par l'instruction XMLGROUP en **CLOB**.

Dans vsc nous obtenons

```
5
6 -- creation d'un arbre <liste_films> en créant des feuilles <dvd> pour chaque ligne de notre table DVD.
7 SELECT XMLGROUP(TRIM(coddvd) AS "code_dvd",
8               TRIM(titre) AS "titre",
9               genre AS "genre"
10              OPTION ROW "dvd" ROOT "liste_Films")
11 FROM yv.dvd;
12 -- transformons le document XML en CLOB big big chaîne de caractère
13 SELECT xmlserialize(XMLGROUP(TRIM(coddvd) AS "code_dvd",
14               TRIM(titre) AS "titre",
15               genre AS "genre"
16              OPTION ROW "dvd" ROOT "liste_Films") as CLOB(10000000) INCLUDING XMLDECLARATION)
17 FROM yv.dvd;
18 -- insérons ce document dans un fichier de l'IFS.
19 CALL QSYS2.IFS_WRITE(
```

PROBLEMS 32 OUTPUT DEBUG CONSOLE TERMINAL GIT LENS IBM I: RESULTS

```
00001
<?xml version="1.0" encoding="UTF-8"?><liste_Films><dvd><code_dvd>10</code_dvd><titre>Star Wars - La menace fantôme</titre><genre>S</genre></dvd><dvd><code_d
Loaded 1. End of data
```

1. creation du fichier xml dans l'IFS

afin de créer directement via sql notre fichier IFS avec le contenu du résultat de notre requête précédente, nous allons utiliser la procédure SQL **IFS_WRITE**

Pour ce faire nous devons préciser :

- **PATH_NAME** => le chemin complet du fichier IFS
- **overwrite** => le mode de gestion ici nous serons en **overwrite** cad annule et remplace du contenu.
- **line** => les lignes à écrire.
- **FILE_CCSID** => le CCSID du fichier résultant, nous choisissons **1208** pour l'UTF-8.

- ajouter cette ligne dans votre script.

⚠️ veuillez à modifier le contenu de la ligne pour préciser le chemin de votre home sur l'IFS.

```
-- insérons ce document dans un fichier de l'IFS.
CALL QSYS2.IFS_WRITE(
  PATH_NAME => '/home/YV/xmlOUT.xml',
  overwrite => 'REPLACE',
  LINE => (
    SELECT xmlserialize(XMLGROUP(TRIM(coddvd) AS "code_dvd",
      TRIM(titre) AS "titre",
      genre AS "genre"
    OPTION ROW "dvd" ROOT "liste_Films") as CLOB(10000000) INCLUDING
XMLDECLARATION)
    FROM yv.dvd),
  FILE_CCSID => 1208);
```

au lancement , nous obtenons .

```
14      TRIM(titre) AS "titre",
15      genre AS "genre"
16  OPTION ROW "dvd" ROOT "liste_Films") as CLOB(10000000) INCLUDING XMLDECLARATION)
17  FROM yv.dvd;
18  -- insérons ce document dans un fichier de l'IFS.
19  CALL QSYS2.IFS_WRITE(
20  .... PATH_NAME => '/home/YV/xmlOUT.xml',
21  .... overwrite => 'REPLACE',
22  .... LINE => (
23  SELECT xmlserialize(XMLGROUP(TRIM(coddvd) AS "code_dvd",
24  .... TRIM(titre) AS "titre",
25  .... genre AS "genre"
26  .... OPTION ROW "dvd" ROOT "liste_Films") as CLOB(10000000) INCLUDING XMLDECLARATION)
27  FROM yv.dvd),
28  .... FILE_CCSID => 1208);
```

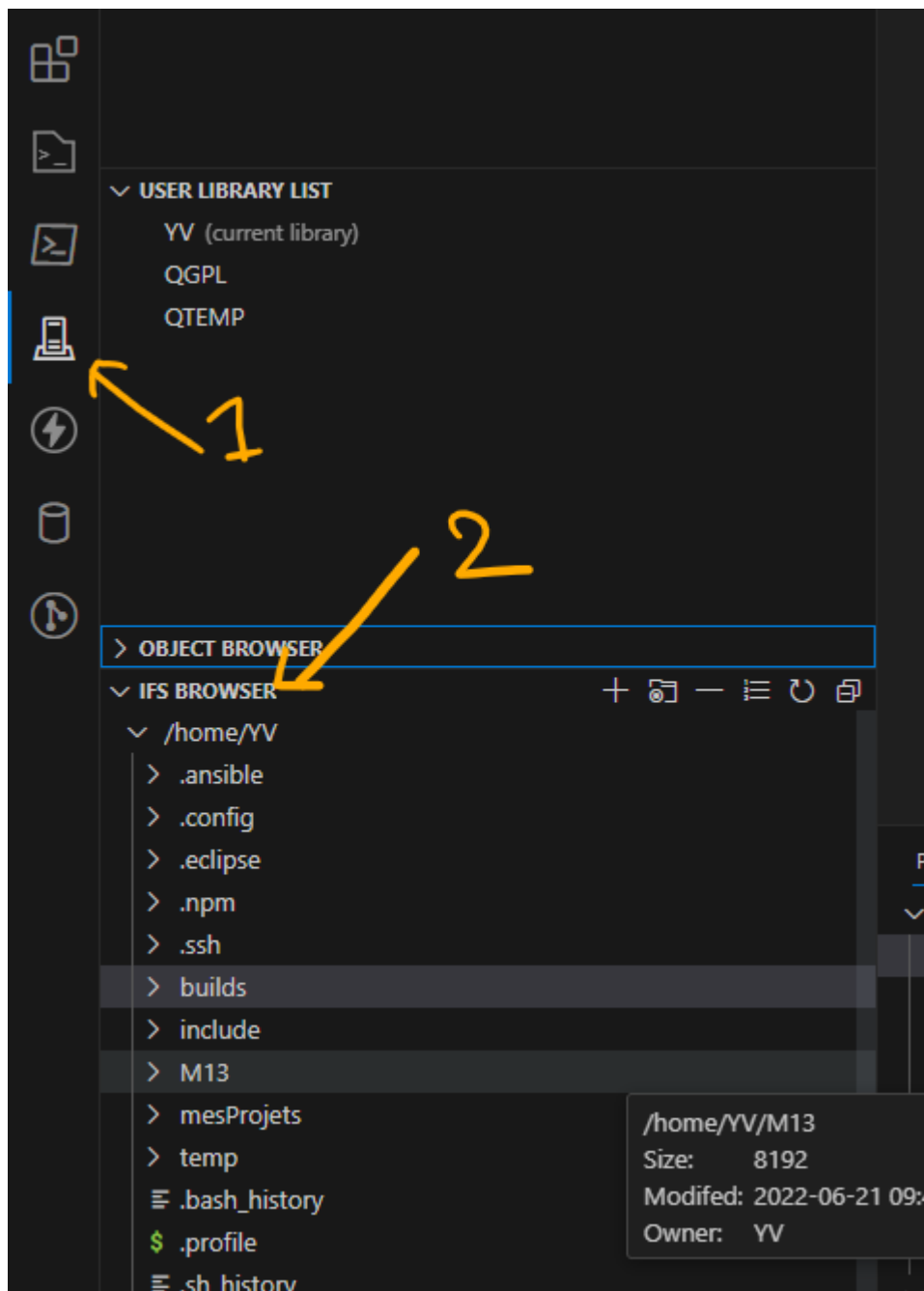
PROBLEMS 32 OUTPUT DEBUG CONSOLE TERMINAL GITLENS IBM I: RESULTS

Query executed with no data returned.

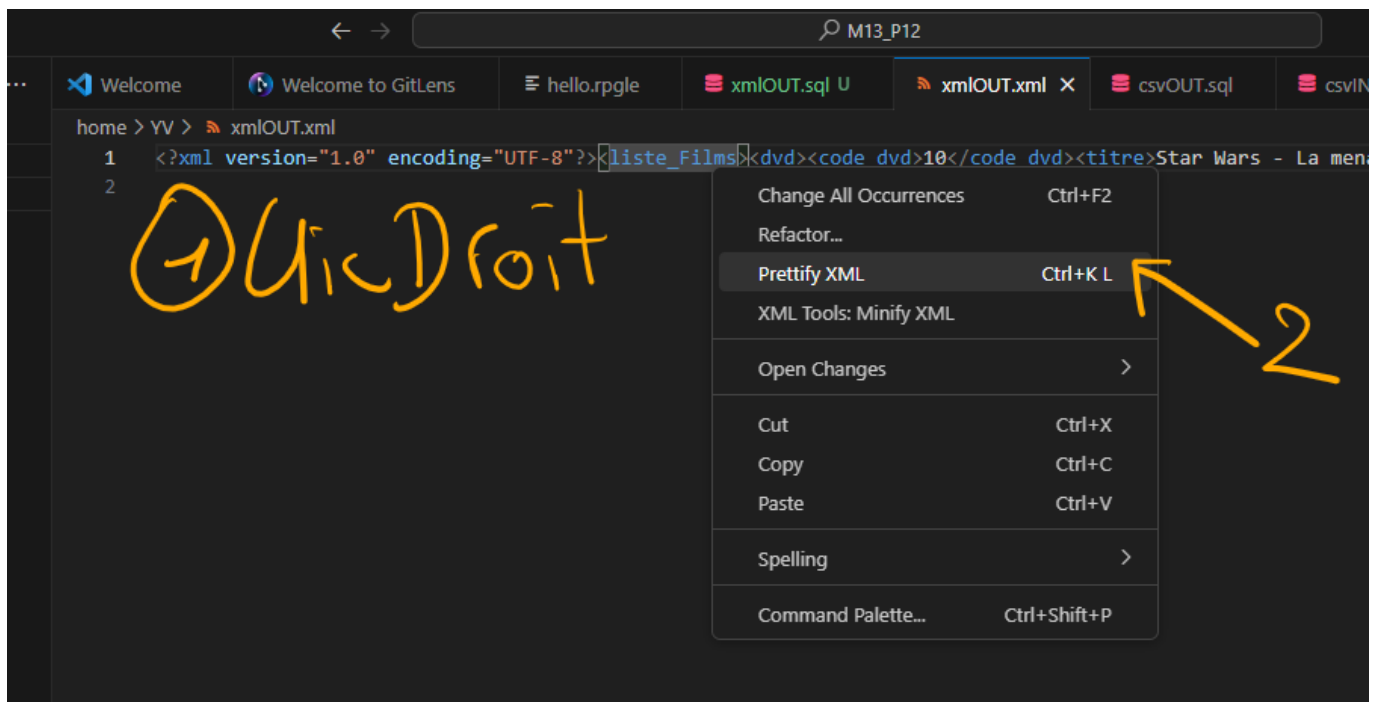
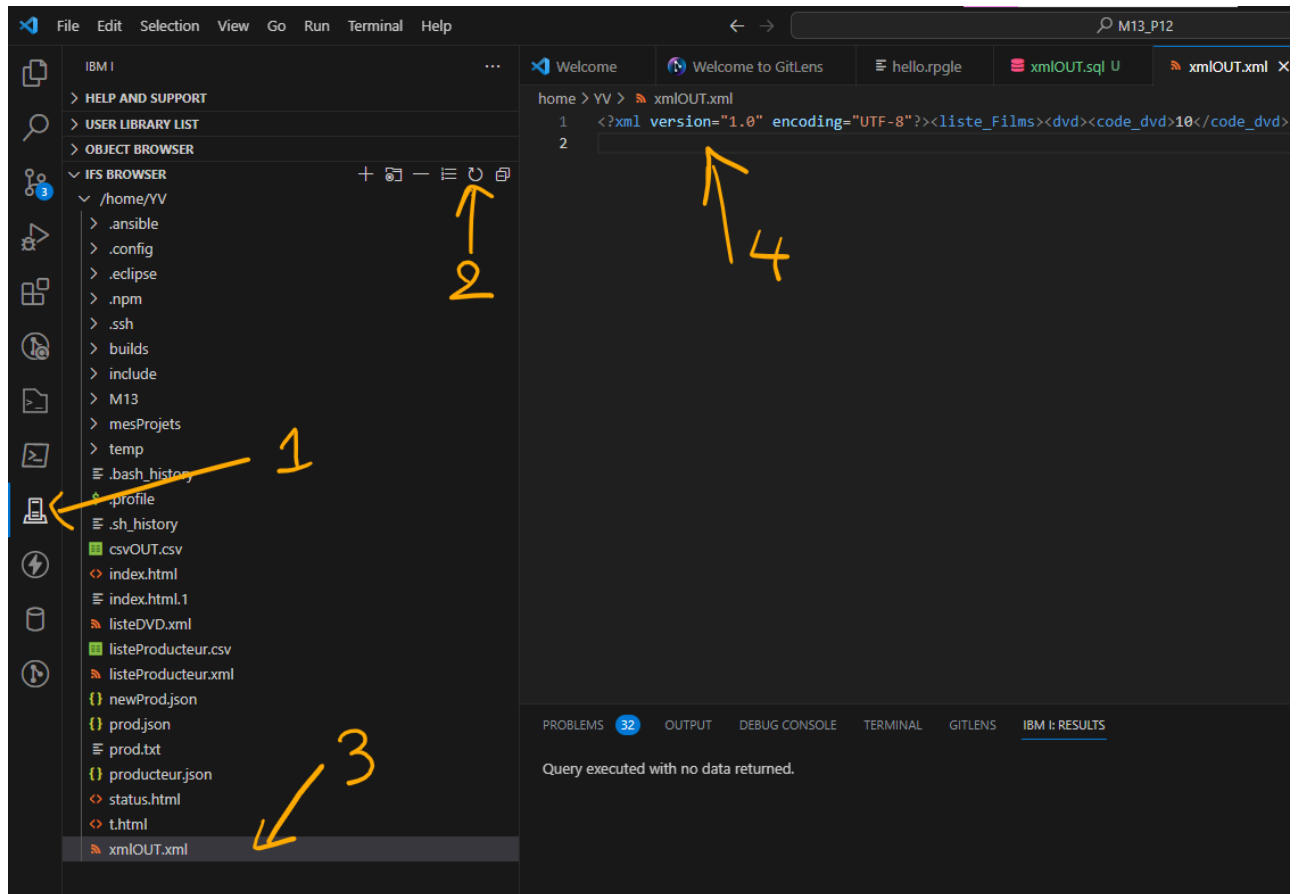
Etape 2 modifier les données

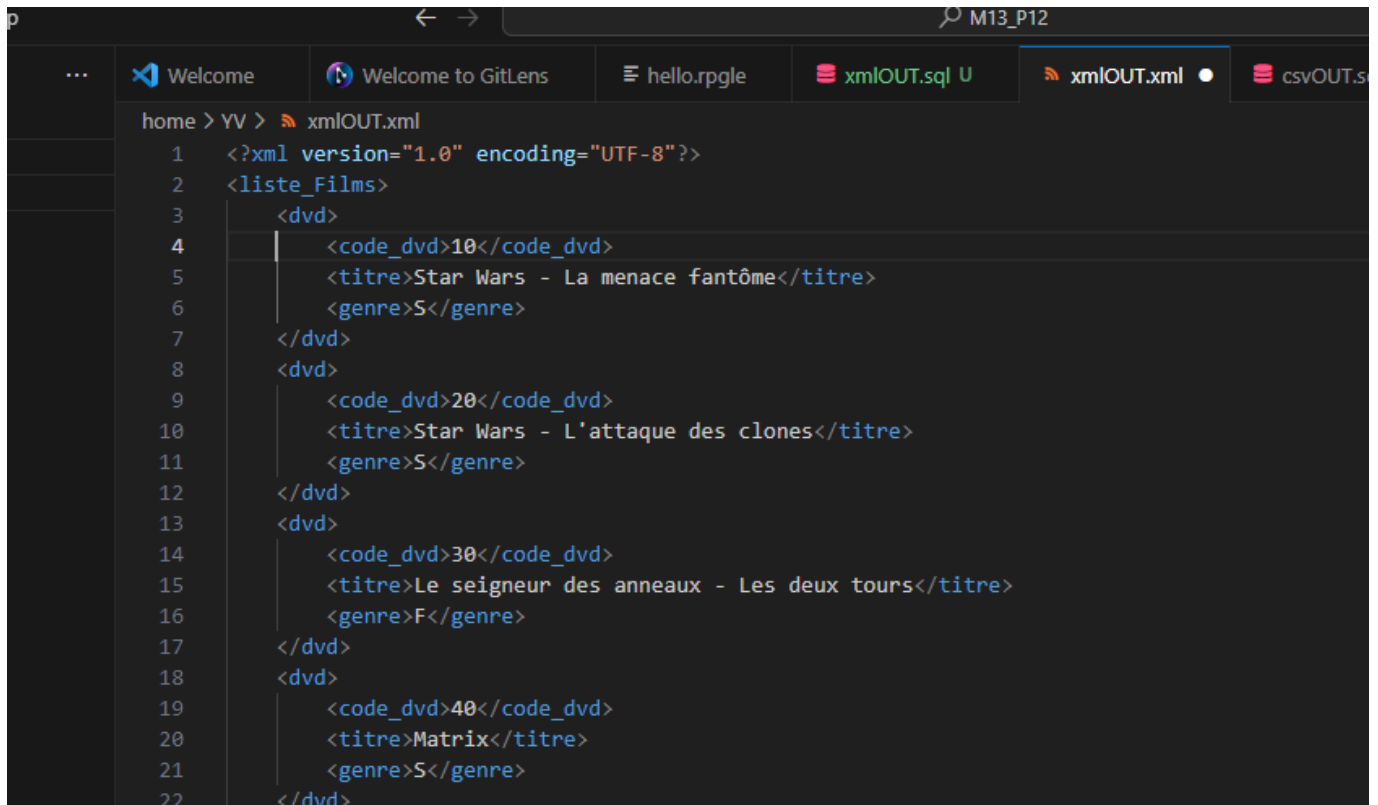
1. visualiser le fichier xml dans VSC

- Ouvrir la vue **IFS BROWSER** dans C4i.



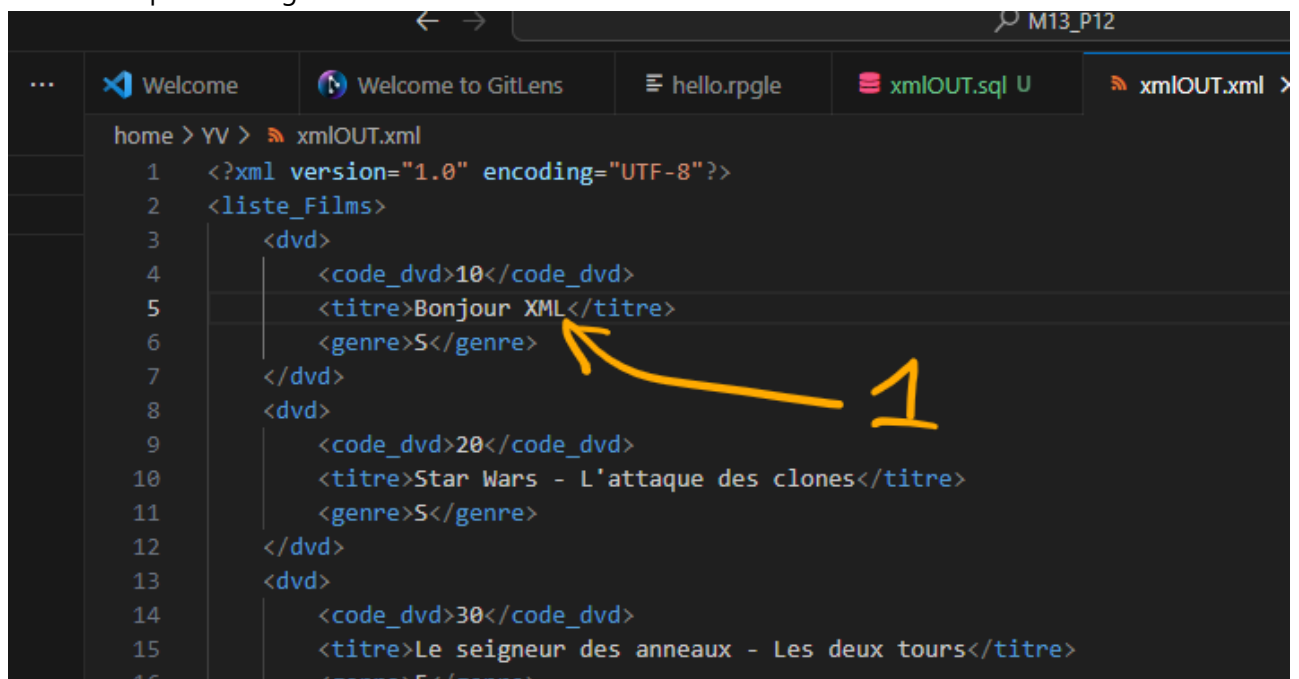
- Ouvrir le fichier xmlOUT.xml





```
home > YV > xmlOUT.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <liste_Films>
3   <dvd>
4     <code_dvd>10</code_dvd>
5     <titre>Star Wars - La menace fantôme</titre>
6     <genre>S</genre>
7   </dvd>
8   <dvd>
9     <code_dvd>20</code_dvd>
10    <titre>Star Wars - L'attaque des clones</titre>
11    <genre>S</genre>
12  </dvd>
13  <dvd>
14    <code_dvd>30</code_dvd>
15    <titre>Le seigneur des anneaux - Les deux tours</titre>
16    <genre>F</genre>
17  </dvd>
18  <dvd>
19    <code_dvd>40</code_dvd>
20    <titre>Matrix</titre>
21    <genre>S</genre>
22  </dvd>
```

- Modifier la première ligne



```
home > YV > xmlOUT.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <liste_Films>
3   <dvd>
4     <code_dvd>10</code_dvd>
5     <titre>Bonjour XML</titre>
6     <genre>S</genre>
7   </dvd>
8   <dvd>
9     <code_dvd>20</code_dvd>
10    <titre>Star Wars - L'attaque des clones</titre>
11    <genre>S</genre>
12  </dvd>
13  <dvd>
14    <code_dvd>30</code_dvd>
15    <titre>Le seigneur des anneaux - Les deux tours</titre>
16    <genre>F</genre>
```

Etape 3 XML IN

1. créer un script sql xmlIN.sql
2. créer un table de travail de la même forme que la table DVD

```
-- Creation de la table de travail.
create table YV.DVDIN as (
  select * from YV.dvd)
with no data;
```


ou vider la table si elle existe déjà.

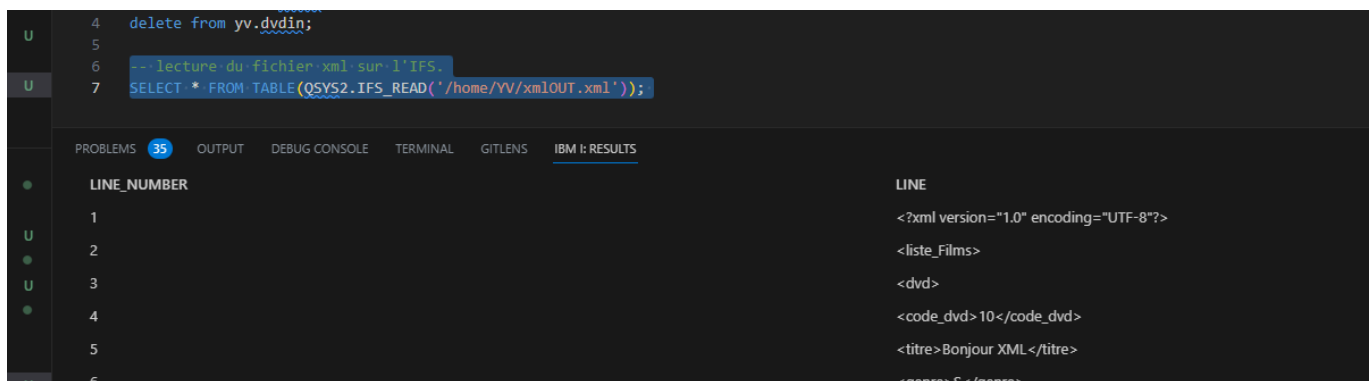
```
delete from YV.DVDIN
```

1. visualiser le contenu du fichier sur l'IFS.

Nous allons utiliser la procédure `IFS_READ` pour afficher le contenu du fichier xml directement dans une pseudo table sql.

- ajouter cette ligne dans votre script.

```
-- lecture du fichier xml sur l'IFS.
SELECT * FROM TABLE(QSYS2.IFS_READ('/home/YV/xmlOUT.xml'));
```



Chaque ligne de notre fichier donne lieu à une ligne dans notre table.

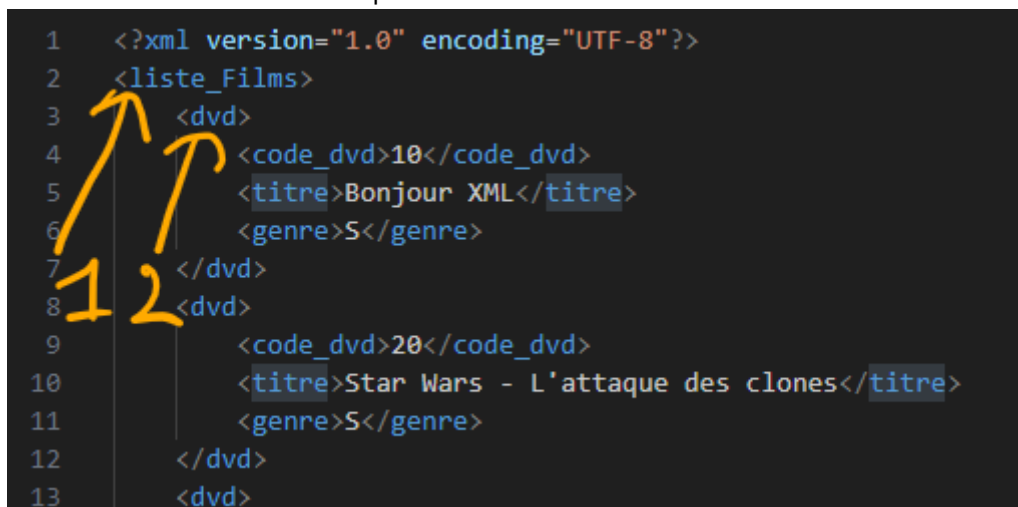
1. intégrer les données dans la table de travail.

Il nous reste à décoder le contenu de la table xml restituée pour en extraire le contenu des balises xml comme des zones d'une table sql.

Pour ce faire nous utilisons ,l'instruction `XMLTABLE`.

Pour ce faire nous devons préciser :

- `'$result/liste_Films/dvd'` précise l'élément de référence pour une ligne de la table dans le document xml le chemin complet du fichier IFS

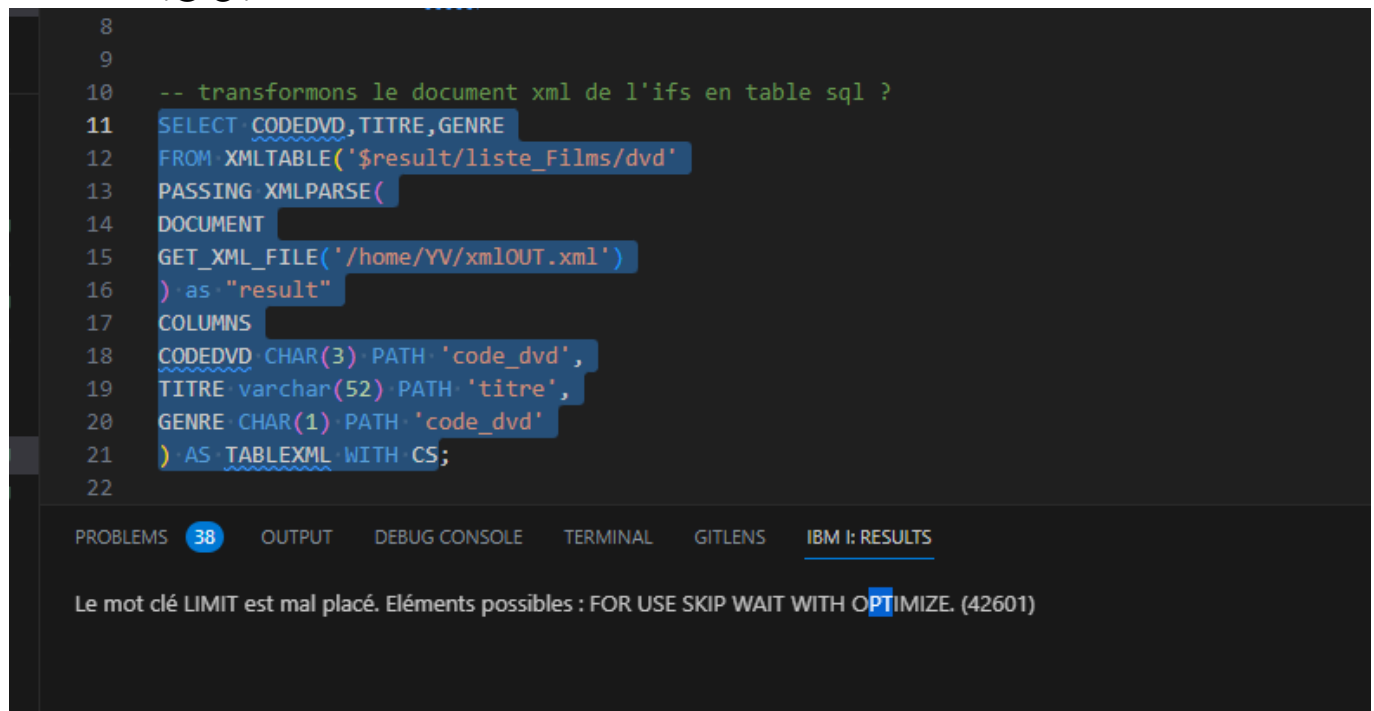


- `PASSING XMLPARSE(DOCUMENT` pour indiquer le document xml.
- `GET_XML_FILE('/home/YV/xmlOUT.xml')` la procédure charge directement le document xml avec le contenu du fichier IFS.

- as "result" précise la racine de notre document notre référence.
- COLUMNS reste à indiquer nos colonnes.
- CODEDVD CHAR(3) PATH 'code_dvd' ma table sql aura une zone code_dvd qui sera un char(3) son contenu sera celui de la balise xml code_dvd ou plus précisément /liste_Films/dvd/code_dvd

```
-- transformons le document xml de l'ifs en table sql ?
SELECT CODEDVD,TITRE,GENRE
FROM XMLTABLE('$result/liste_Films/dvd'
PASSING XMLPARSE(
DOCUMENT
GET_XML_FILE('/home/YV/xmlOUT.xml')
) as "result"
COLUMNS
CODEDVD CHAR(3) PATH 'code_dvd',
TITRE varchar(52) PATH 'titre',
GENRE CHAR(1) PATH 'code_dvd'
) AS TABLEXML WITH CS;
```

ko dans vsc (🤔💣)



```
8
9
10 -- transformons le document xml de l'ifs en table sql ?
11 SELECT CODEDVD,TITRE,GENRE
12 FROM XMLTABLE('$result/liste_Films/dvd'
13 PASSING XMLPARSE(
14 DOCUMENT
15 GET_XML_FILE('/home/YV/xmlOUT.xml')
16 ) as "result"
17 COLUMNS
18 CODEDVD CHAR(3) PATH 'code_dvd',
19 TITRE varchar(52) PATH 'titre',
20 GENRE CHAR(1) PATH 'code_dvd'
21 ) AS TABLEXML WITH CS;
22
```

PROBLEMS 38 OUTPUT DEBUG CONSOLE TERMINAL GITLENS IBM I: RESULTS

Le mot clé LIMIT est mal placé. Éléments possibles : FOR USE SKIP WAIT WITH OPTIMIZE. (42601)

mais si vous lancer la requête dans acs vous avez...

```

1  -- transformons le document xml de l'ifs en table sql ?
2  SELECT CODEDVD,TITRE,GENRE
3  FROM XMLTABLE('$result/liste_Films/dvd'
4  PASSING XMLPARSE(
5  DOCUMENT
6  GET_XML_FILE('/home/YV/xmlOUT.xml')
7  ) as "result"
8  COLUMNS
9  CODEDVD CHAR(3) PATH 'code_dvd',
10 TITRE varchar(52) PATH 'titre',
11 GENRE CHAR(1) PATH 'code_dvd'
12 ) AS TABLEXML WITH CS;

```

CODEDVD	TITRE	GENRE
10	Bonjour XML	1
20	Star Wars - L'attaque des clones	2
30	Le seigneur des anneaux - Les deux tours	3
40	Matrix	4
50	Spiderman	5
60	Jurassik Park	6
70	Jurassik Park - Le monde perdu	7
80	Jurassik Park III	8

PROBLEMS

38

OUTPUT

DEBUG CONSOLE

TERMINAL

GITLENS

IBM I: RESULTS

Les pointeurs LOB et XML ne sont pas admis avec COMMIT(*NONE). (42926)

Noter que nous obtenons une véritable table sql et qu'à ce titre onus pouvons utiliser toute la puissance du sql pour filter,ordonner, modifier les données du fichier xml (enfin sa représentation) . taper la ligne)

```

-- transformons le document xml de l'ifs en table sql ?
SELECT CODEDVD,LCASE(TITRE),GENRE
FROM XMLTABLE('$result/liste_Films/dvd'
PASSING XMLPARSE(
DOCUMENT
GET_XML_FILE('/home/YV/xmlOUT.xml')
) as "result"
COLUMNS
CODEDVD CHAR(3) PATH 'code_dvd',
TITRE varchar(52) PATH 'titre',
GENRE CHAR(1) PATH 'code_dvd'
) AS TABLEXML where GENRE ='1' order by TITRE WITH CS;

```

vous obtenez

```

1  -- transformons le document xml de l'ifs en table sql ?
2  SELECT CODEDVD,LCASE(TITRE),GENRE
3  FROM XMLTABLE('$result/liste_Films/dvd'
4  PASSING XMLPARSE(
5  DOCUMENT
6  GET_XML_FILE('/home/YV/xmlOUT.xml')
7  ) as "result"
8  COLUMNS
9  CODEDVD CHAR(3) PATH 'code_dvd',
10 TITRE varchar(52) PATH 'titre',
11 GENRE CHAR(1) PATH 'code_dvd'
12 ) AS TABLEXML where GENRE ='1' order by TITRE WITH CS;

```

CODEDVD		GENRE
10	bonjour xml	1
130	le seigneur des anneaux - la communauté	1
140	le seigneur des anneaux - retour du roi	1
150	matrix reloaded	1
160	matrix révolutions	1
110	star wars - l'empire contre-attaque	1
120	star wars - le retour du jedi	1
100	star wars - un nouvel espoir	1
170	terminator	1
180	terminator 2 - le jugement dernier	1
190	terminator 3 - soulèvement des machines	1

Intégrer la lecture du documentXML dans un programme RPG.

1. copier le source du programme SQL21FREEA dans dspXML.sqlrpgle via ctrl-c puis ctr-v du contenu (c'est plus simple..)

```

1  **free
2  // Exercice 21 - Sous-fichier statique - Partie A
3
4  dcl-f dsp21a workstn sfile(fmtenr:rrn);
5  dcl-s rrn zoned(4);
6  exec sql set option COMMIT = *NONE , DATFMT=*ISO;
7  exec sql
8      declare listeFilms cursor for
9          select titre,annee from dvd order by titre
10         for fetch only;
11  exsr remplissageSfl;
12  exsr affichagePage1;
13
14  *inlr = *on;
15
16  begsr remplissageSfl;
17      exec sql
18          open listeFilms;
19      dow not %eof;
20          exec sql
21              fetch from listeFilms into :titre,:annee;
22              rrn = rrn + 1;
23              write fmtenr;
24      enddo;
25  exec sql
26      close listeFilms;
27  *in34 = *on;
28  endsr;
29
30  begsr affichagePage1;
31      write touches;
32      exfmt sfct1;
33  endsr;

```

2. compiler et tester ctrl-e puis

```

src > qrpglesrc > dspXml.sqlrpgle
1  **free
2  // Exercice 21 - Sous-fichier statique - Partie A
3
4  dcl-f dsp21a workstn sfile(fmtenr:rrn);
5  dcl-s rrn zoned(4);
6  exec sql set option COMMIT = *NONE , DATFMT=*ISO;
7  exec sql
8      declare listeFilms cursor for
9          select titre,annee from dvd order by titre

```

sur une session 5250 lancer le programme

```

Liste des Films

titre du Film :                               Année :
Batman                                             1989
Batman Forever                                   1995
Batman Le Défi                                   1992
Bienvenue à Gattaca                             1997
Blade Runner                                    1992
Demolition Man                                  1993
Harry Potter et le prisonnier d'Azkaban          2004
Hollow Man                                       2000
I Robot                                          2004
Jurassik Park                                   1993

F3=Exit
À suivre...

```

3. ajouter le control de validation par l'option en début de programme

```
exec sql SET OPTION COMMIT=*CS,CLOSESQLCSR=*ENDMOD,DATFMT=*ISO ;
```

1. changer la requête du curseur.

```
exec sql
  declare listeFilms cursor for
  select titre,annee from dvd order by titre
  for fetch only;
```

devient

```
exec sql
  declare listeFilms cursor for
  SELECT TITRE
  FROM XMLTABLE('$result/liste_Films/dvd'
  PASSING XMLPARSE(
  DOCUMENT
  GET_XML_FILE('/home/YV/xmlOUT.xml')
  ) as "result"
  COLUMNS
  TITRE varchar(52) PATH 'titre'
  ) AS TABLEXML order by titre
  for fetch only;
```

penser à modifier le code de la boucle

```
DoU 1 = 0;
exec sql
  fetch from listeFilms into :titre;
  if sqlcode < *zeros or sqlcode = 100;
    leave;
  endif;
```

Nous avons enlevé l'année. 🤖 j'ai oublié de la mettre dans le document xml mais le principe est là.

1. compiler et tester cool non ?



Conclusion et feed-back

Correction

💡💡💡💡 Idées

•