RPG-REGEX-JAVA

- Les articles -

Date de mise en ligne : samedi 5 novembre 2005

Description :

Programme de service REGEXSRV (*SRVPGM) permettant d'utiliser facilement les "Expressions Régulières" à partir de programmes RPG (ou autres...).

Environnement iSeries

Utilisez la puissance des REGEX à partir du RPG !

Programme de service REGEXSRV (*SRVPGM) permettant d'utiliser les "Expressions Régulières" à partir de programmes RPG (ou autres...).

Vous trouverez beaucoup de site sur le Net traitant des expressions régulières et je ne vais pas m'attarder, dans cet article, sur le sujet.

Les "Expressions Régulières" permettent de vérifier que la phrase à examiner respecte un motif ou une série de motifs (nous utiliserons le terme "pattern") désignant la syntaxe attendue de la phrase.

Nous utiliserons les Regex dans cet exemple pour :
— Vérifier la syntaxe d'une chaîne de caractères (ex : adresse email de forme xxxxxx@domaine.fr) à partir d'un pattern (par exemple "[A-Z0-9._%-]+@[A-Z0-9._%-]+\.[A-Z]{2,4}" pour vérifier la validité d'une adresse mail).
— Rechercher des occurrences de chaînes spécifiques (à partir d'un pattern) et de les remplacer par une autre chaîne.

- Pour cela vous devrez utiliser un ensemble d'API opensource JAVA nommées REGEXP et téléchargable sur le site http://jakarta.apache.org/regexp (Apache Software Foundation).

- Le programme de service utilise 2 méthodes de l'objet RE :
— match qui permet de trouver l'occurence d'un pattern dans une chaine de caractères.

— subst qui permet de remplacer une ou toutes les occurences d'un pattern dans une chaine de caractère.

- Voici les prototypes du constructeur de la classe RE et des 2 méthodes match et subst :

```
 * public org.apache.regexp.RE(java.lang.String,int)      D RE...      D                 pr             O   ExtProc(*Java :      D                              'org.apache.regexp.RE' :      D
*Constructor)      D                              o   Class(*JAVA :'java.lang.String')      D matchFlags                 10I 0 Value      *aboolean match(java.lang.String)      D match...      D                 pr
    N   ExtProc(*Java :      D                              'org.apache.regexp.RE' :      D                              'match')      D matcher                   O   Class(*Java :'java.lang.String')      *String
subst(String substituteIn, String substitution, int flags)      D re_subst...      D                 pr             O   ExtProc(*Java :      D                              'org.apache.regexp.RE' :      D
        'subst')      D                              Class(*Java :'java.lang.String')      D matcher                   o   Class(*JAVA :'java.lang.String')      D replace                   o   Class(*JAVA
:'java.lang.String')      D substFlags                 10I 0 Value
```

Programme de service REGSRV

Le programme de service ne contient qu'une procédure Re_MATCH qui renvoie une valeur booléenne (*OFF=aucune occurence ou *ON= occurence trouvée).

Afin de s'assurer que la mémoire sera libérer quand les objets java ne sont plus utilisés REGSRV s'appui sur un autre programme de service JVMSRV qui devra être lié à la compilation.

- Prototype de REGSRV : J'ai inclu dans le prototype la définition de certaines constantes (j'ai utilisé les mêmes noms que ceux de l'API java).

```
/IF DEFINED(REGEXSRV)      /EOF      /ENDIF      /DEFINE REGEXSRV      *aConstante pour matchFlags      DMATCH_NORMAL...      D                c                const(x'0000')
DMATCH_CASEINDEPENDENT...      D                c                const(x'0001')      DMATCH_MULTILINE...      d                c                const(x'0002')      dMATCH_SINGLELINE...      d
  c                const(x'0004')      dREPLACE_ALL...      d                c                const(X'0000')      dREPLACE_FIRSTONLY...      d                c                const(X'0001')
dREPLACE_BACKREFERENCES...      d                c                const(X'0002')

d      DRe_MATCH        pr             n                              *on Trouvé      D W_pattern                  255    VALUE varying                    pattern      D Ptr_matcher                  *   VALUE
                  chaine recherchée      dmatchFlags                  10I 0 VALUE OPTIONS(*NOPASS)      d W_replace                  255A   VALUE options(*nopass) varying      dsubstFlags
        10I 0 value options(*nopass)
```

Le programme de service réagit en fonction du nombre de paramètres transmis.

- Si le 4ème paramètre n'est pas renseigné alors on effectue une simple recherche et la procédure renvoie *ON si elle trouve une occurence du pattern recherché.

- Si le 4ème paramètre (et éventuellement le 5ème) est renseigné on effectue une recherche et un remplacement (si la recherche aboutit). La procédure renvoie *ON si elle trouve une occurence du pattern recherché et remplaces le (ou les) occurences trouvées en utilisant le pointeur pour modifier la chaine d'origine.

- Description des paramètres

— W_pattern motif utilisé par la REGEX
— Ptr_matcher un pointeur permettant de transmettre la chaine de caractères à traiter et éventuellement effectuer les remplacements.
— matchFlags un entier pouvant prendre plusieurs valeurs pour stipuler si la recherche se fera en respectant la casse, prendre en compte toutes les lignes... Les constantes MATCH_xxxxxx peuvent être utilisées. Par exemple matchFlags = MATCH_CASEINDEPENDENT pour ignorer la casse. Si le paramètre n'est pas renseigné il prend la valeur MATCH_NORMAL
— W_replace une chaine de caractère. Ce paramètre optionnel (il n'est pas renseigné pour une simple recherche) correspond à la chaine de remplacement.
— substFlags un entier pouvant prendre plusieurs valeurs pour stipuler par exemple si toutes les occurences trouvées doivent être remplacées... Les constantes REPLACE_xxxx peuvent être utilisées. Par exemple substFlags = REPLACE_FIRSTONLY pour ne remplacer que la 1ère occurence de la chaine. Si le paramètre n'est pas renseigné il prend la valeur REPLACE_ALL.

Exemple d'utilisation REGEXSRV avec RPGLE

```
 * Créer  module (opt 15 PDM) + CRTPGM PGM(BIB/REGEXEX) BNDSRVPGM(REGEXSRV)      h copyright('Serge GOMES')      H Thread(*Serialize)      H Option( *SrcStmt)      /COPY
SERGE/PROTOTYPE,REGEXSRV      *a corps de la fonction      D W_pattern      s            255A   varying      DW_matcher      s            32700    varying      D Ptr_matcher      s             *
inz(%addr(W_matcher))      dmatchFlags      s            10I 0 inz      d W_replace      s            255A   varying      dsubstFlags      s            10I 0 inz      dW_ok            s              N   inz      * Le
caractère d'échappement en REGEX est "\"      dW_ESC           c                const(x'48')      * ----------------------------------------------------------------------------------------Euros*      /free
W_pattern = 'M'+W_ESC+'.|MR' ;              // Remplace dans W_MATCHER "M." ou "MR" par "Monsieur" pattern = 'M\.|Mr'                W_matcher = 'Mr Dupont et M. Dupond sont 2
personnages...' ;                W_replace = 'Monsieur' ;                // non respect la casse pour la recherche                matchFlags = MATCH_CASEINDEPENDENT ;                W_ok      =
re_match(W_pattern:Ptr_matcher:matchFlags :                                  W_replace:substFlags) ;                // Maintenant W_matcher vaut                // 'Monsieur Dupont et Monsieur Dupond
sont 2 personnages...'


 // Recherche dans W_MATCHER M. ou Mr en respectant la casse (valeur par défaut)                W_matcher = 'Mr Dupont et m. Dupond sont 2 personnages...' ;                W_ok      =
re_match(W_pattern:Ptr_matcher) ;                // W_OK = *OFF occurence non trouvée                *inLR = *on ;       /end-free
```

Attention les caractères spéciaux utilisés dans les "patterns" peuvent être plus ou moins bien interprétés en fonction du CCSID de votre machine. Le caractère "\" par exemple peut être interprété "/", si vous avez un doute utilisez la valeur hexa du caractère.

Installation :

- Téléchargez tous les codes sources et mettez les dans vos fichiers sources correspondants (QRPG..., PROTOTYPE.., membre regsrv et jvmsrv).

Pensez à remplacerer les lignes /copy dans REGSRV et JVMSRV par vos propres chemins.

- créez un fichier source QSRVSRC dans votre bibliothèque source(CRTSRCPF BIB/QSRVSRC). Créez 2 nouveaux membres et insérer les script de liage (BND) suivants :

— JVMSRV

```
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('JVMSRV_V01')
 EXPORT SYMBOL("GETJNIENV")
 EXPORT SYMBOL("FREEJAVAOBJECT")
 ENDPGMEXP
```

— REGEXSRV

```
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('REGEXSRV_V01')
 EXPORT SYMBOL("RE_MATCH")
 ENDPGMEXP
```

- Compilez jvmsrv (procédez en 2 temps création du module puis CRTSRVPGM)
- Compilez regsrv procédez en 2 temps et n'oubliez pas de lier jvmsrv à la compilation :

```
CRTSRVPGM SRVPGM(BIB/REGEXSRV)
 MODULE(BIB/REGEXSRV)
 EXPORT(*SRCFILE)
 BNDSRVPGM(BIB/JVMSRC)
```

- A l'exécution la variable d'environnement CLASSPATH doit être initialisée de manière à pointer sur l'emplacement des classes Java (ex :'/QIBM/JAVA/jakarta-regexp-1.3.jar'). Vous pouvez utiliser la commande WRKENVAR ou ADDENVAR pour régler cette valeur.