Modernisation, développement d'applications et DB2 sous IBM i

Technologies, outils et nouveautés 2012-2013

8 et 9 avril 2013 – IBM Client Center Paris, Bois-Colombes

S12 – Profitez des fonctionnalités JAVA dans vos programmes RPG

Lundi 9 avril – 9h00-10h30

Nathanaël BONNET – GAIA Mini Systèmes

JAVA

- Langage de programmation Orienté Objet
  - RPG est un langage procédural
- Langage extrêmement implanté
  - Un des standards du marché, avec COBOL, C, .Net
  - Open Source, gratuit
- Langage portable
  - Ce n'est pas le seul, avec des conditions (JNI) …
  - La portabilité repose sur le principe de machine virtuelle
    - Le code Java compilé ne produit pas un programme exécutable
      - Obligatoirement tributaire d'une plateforme
    - Mais un « byte code »
      - Un code pseudo-compilé, qui peut fonctionner sur une machine virtuelle
    - Chaque plateforme dispose d'implémentations de la machine virtuelle
      - C'est un logiciel qui émule un environnement d'exécution pour le byte code

Que peut m'apporter JAVA

- Des fonctionnalités absentes de RPG !
- La communauté JAVA est importante
  - De nombreux codes sources libres et gratuits sont disponibles
- Utiliser les standards du marché
  - Tous les éditeurs fournissent des connecteurs JAVA
    - Se connecter à n'importe quelle BD …
  - Toutes les normes actuelles
    - XML, XSL, XSLT, HTML, WebServices, SOAP …
  - Des fonctionnalités plus particulières
    - Calcul scientifique, financier …
- Gain de productivité

JAVA sur IBM i – Prérequis

- JAVA est fourni avec l'OS
  - Produit JV1

5761-JV1	base	IBM Developer Kit for Java™
5761-JV1	8	J2SE 5.0 32 bits
5761-JV1	9	J2SE 5.0 64 bits
5761-JV1	11	Java SE 6 32 bits
5761-JV1	12	Java SE 6 64 bits
5761-JV1	13	J2SE 1.4 64 bits

- Les nouvelles versions de la JVM nécessitent PASE
  - Les seules versions supportées en 7.1
  - Portable Application Solution Environment
    - Supporte les binaires AIX
  - Produit SS1 option 33

JAVA sur IBM i – Prérequis

- Il existe deux types de JVM
  - IBM Technology for Java (IT4J)
    - JVM « Power Systems »
      - Utilisée également par AIX (cf. PASE)
  - Classic
  - Nous ne parlerons que de la 1ère, qui est recommandée. La Classic est appelée à disparaitre
- Le support de la JVM permet
  - L'exécution de code Java
    - En ligne de commande
    - Depuis RPG
    - Dans un serveur d'applications
      - WebSphere, IWS, TomCat …

JAVA depuis RPG

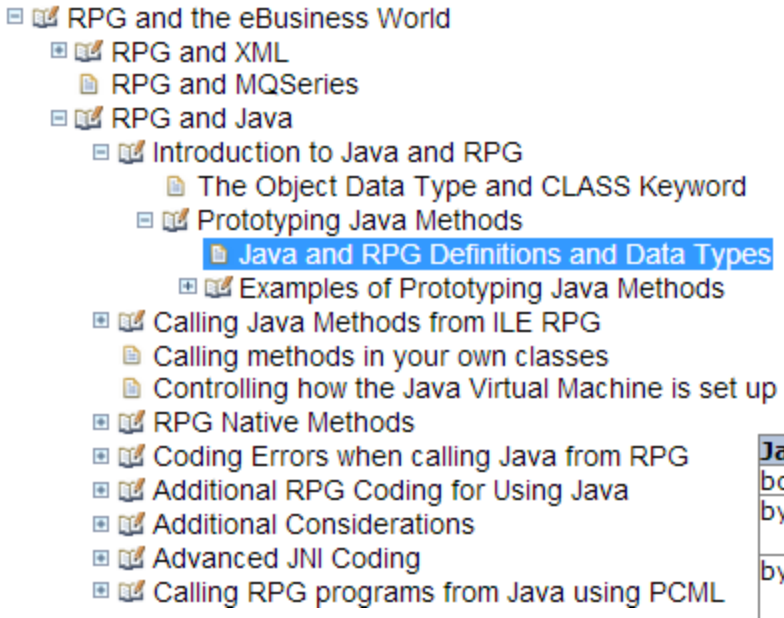
- JAVA est orienté Objet
  - Classe et instance
    - Attribut
    - Méthode
  - Héritage
  - Encapsulation
  - Polymorphisme
- RPG ILE est procédural
  - Sait appeler des procédures d'autres langages
    - C, C++, COBOL, CL, JAVA
      - Pour Java, une méthode ⇔ une procédure
  - Nécessite un prototype

JAVA depuis RPG

- Pour prototyper un appel de méthode JAVA
  - Nom complet de la classe (packages et sous-packages)
  - Nom de la méthode
  - Type des paramètres
  - Type de la valeur de retour
  - Méthode de classe (static) ou d'instance
  - Constructeur ?
- Réaliser l'équivalence de type JAVA ⇔ RPG

JAVA depuis RPG

- Infocentre



Java Data Type	ILE RPG Data Type	RPG Definitions
boolean	indicator	N
byte [1]	integer	3I 0
	character	1A
byte[]	character length > 1 (See 3.)	nA
	array of character length=1 (See 4.)	1A DIM(x)
	date	D
	time	T
	timestamp	Z
short	2-byte integer	5I 0
char	UCS-2 length=1	1C
char[]	UCS-2 length>1 (See 3.)	nC
	array of UCS-2 length=1 (See 4.)	1C DIM(x)
int	4-byte integer	10I 0
long	8-byte integer	20I 0
float	4-byte float	4F
double	8-byte float	8F
any object	object	O CLASS(x)
any array	array of equivalent type (See 4.)	DIM(x)

Exemple

- Utilisons la classe `String`
  - Elle dispose d'une méthode `equalsIgnoreCase`
    - Permet de comparer 2 chaînes de caractères sans tenir compte de la casse

- JAVA

```
// Déclarations
String s1, s2;

// Instanciation
s1 = new String("Valeur à comparer");
s2 = new String("VaLeUr à CoMpArEr");
// Comparaison
if (s1.equalsIgnoreCase(s2))
    System.out.println("Chaines identiques");
else
    System.out.println("Chaines différentes");
```

Equivalent carte D

Appel au constructeur
Nécessite un prototype

Appel à la méthode
Nécessite un prototype

```
<arrêté> Exemple1 [Application Java]
Chaines identiques
```

Exemple

- RPG
  - La JavaDoc fournit les informations nécessaires pour réaliser les prototypes

Class String

java.lang.Object
  java.lang.String

String

```
public String(byte[] bytes)
```

```
 // Constante nom de la classe
d class_String     c                   const( 'java.lang.String' )
```

```
 // Constructeur
d str_byte_new     pr                  o   extproc( *java
d                                          : class_String
d                                          : *constructor )
d                                          class( *java : class_String )
d  string                          65535a  const options( *varsize )
```

equalsIgnoreCase

```
public boolean equalsIgnoreCase(String anotherString)
```

```
 // Méthode equalsIgnoreCase
d str_equalsIgnoreCase...
d                  pr                  n   extproc( *java
d                                          : class_String
d                                          : 'equalsIgnoreCase' )
d  anotherString                       o   class( *java : class_String )
d                                          const
```

Exemple

```
 // Variables
 // ---------
d s1              s                   o   class( *java : class_String )
d s2              s                   o   class( *java : class_String )

 /free

  // instanciation
  s1 = str_byte_new( 'Valeur à comparer' ) ;
  s2 = str_byte_new( 'VaLeUr à CoMpArEr' ) ;

  // Comparaison
  if ( str_equalsIgnoreCase( s1 : s2 ) ) ;
     dsply 'Chaines identiques' ;
  else ;
     dsply 'Chaines différentes' ;
  endif ;
```

Déclaration de variables RPG de type Objet

Appels du constructeur

Appel à la méthode `equalsIgnoreCase`
Le 1er paramètre est implicitement l'instance

- JAVA : notation pointée
  - `s1.equalsIgnoreCase(s2)`
- RPG : paramètre implicite en 1ère position
  - `str_equalsIgnoreCase( s1 : s2 )`

```
> call s12_1
  La machine JVM est IBM Technology for Java. PID (138170)
  DSPLY  Chaines identiques
```

Remarques

- L'utilisation dans le RPG n'est pas difficile
- La difficulté réside dans la création des prototypes
  - Avoir des bases en JAVA
  - Maîtriser l'ILE
  - Trouver la documentation JAVA
  - Réaliser les équivalences avec RPG
  - ...
- Plus on utilise d'objets et de méthodes de classes différentes
  - Plus le nombre de prototypes sera important

Un exemple concret

- Proposer un correcteur orthographique dans vos écrans 5250 !
- Google propose un correcteur
  - Gratuit, en ligne (aucune installation)
  - https://code.google.com/p/google-api-spelling-java/

Google Spell Check

google-api-spelling-java

A Java API for Google spell checking service

Project Home | Downloads | Wiki | Issues | Source

Summary | People

Project Information

+1 +3 Recommend this on Google

Starred by 30 users
Project feeds

Code license
Apache License 2.0

Labels
spelling, kamran, api, java, client, service, web, xeusman

Members
xeus....@gmail.com

Featured

Downloads
google-api-spelling-java-1.1-src.tgz
google-api-spelling-java-1.1-with-dependencies.zip
google-api-spelling-java-1.1.jar
Show all »

Links

Blogs
Kamran

This project will soon be migrated to Github

Introduction

This is a simple Java API that makes it very easy to call Google's spell checker servic

How to use it?

Below is a simple example that demonstrate it's usage.

```
SpellChecker checker = new SpellChecker();

SpellResponse spellResponse = checker.check( "helloo worlrd" );

for( SpellCorrection sc : spellResponse.getCorrections() )
   System.out.println( sc.getValue() );
```

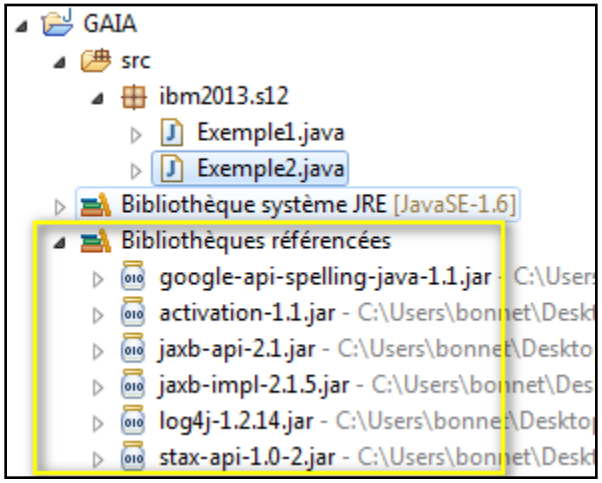
This will print all the corrections available for the text "helloo worlrd".

It is also possible to set more options to the request, here is a meatier example.

```
// Proxy settings
Configuration config = new Configuration();
config.setProxy( "my_proxy_host", 8080, "http" );

SpellChecker checker = new SpellChecker( config );
checker.setOverHttps( true ); // Use https (default true from v1.1)
checker.setLanguage( Language.ENGLISH ); // Use English (default)

SpellRequest request = new SpellRequest();
request.setText( "helloo helloo worlrd" );
request.setIgnoreDuplicates( true ); // Ignore duplicates

SpellResponse spellResponse = checker.check( request );

for( SpellCorrection sc : spellResponse.getCorrections() )
   System.out.println( sc.getValue() );
```

Appel JAVA

```
// Paramètrage proxy
Configuration config = new Configuration();
config.setProxy( "proxy", 8080, "http" );

SpellChecker checker = new SpellChecker( config );
checker.setOverHttps( true );            // Communication https
checker.setLanguage( Language.FRENCH ); // English par défaut

SpellRequest request = new SpellRequest();
request.setText( "Bienven dans cete session RPG et JAVA" );
// request.setIgnoreDuplicates( true );

// Appel au correcteur sur le net
SpellResponse spellResponse = checker.check( request );

// Lister les réponses
for( SpellCorrection sc : spellResponse.getCorrections() )
   System.out.println( sc.getValue() );
```




```
<arrêté> Exemple2 [Application Java] C:\Program Files (x86)\IBM\Installation Man
[DEBUG] [org.xeustechnologies.googleapi.spelling.SpellC
[DEBUG] [org.xeustechnologies.googleapi.spelling.SpellC
Bienvenue       Brava    Bava     Bravas   Bavas    Pava
cette    ce te    ce-te    celte    ceste
```

Fonctionnement « under the cover »

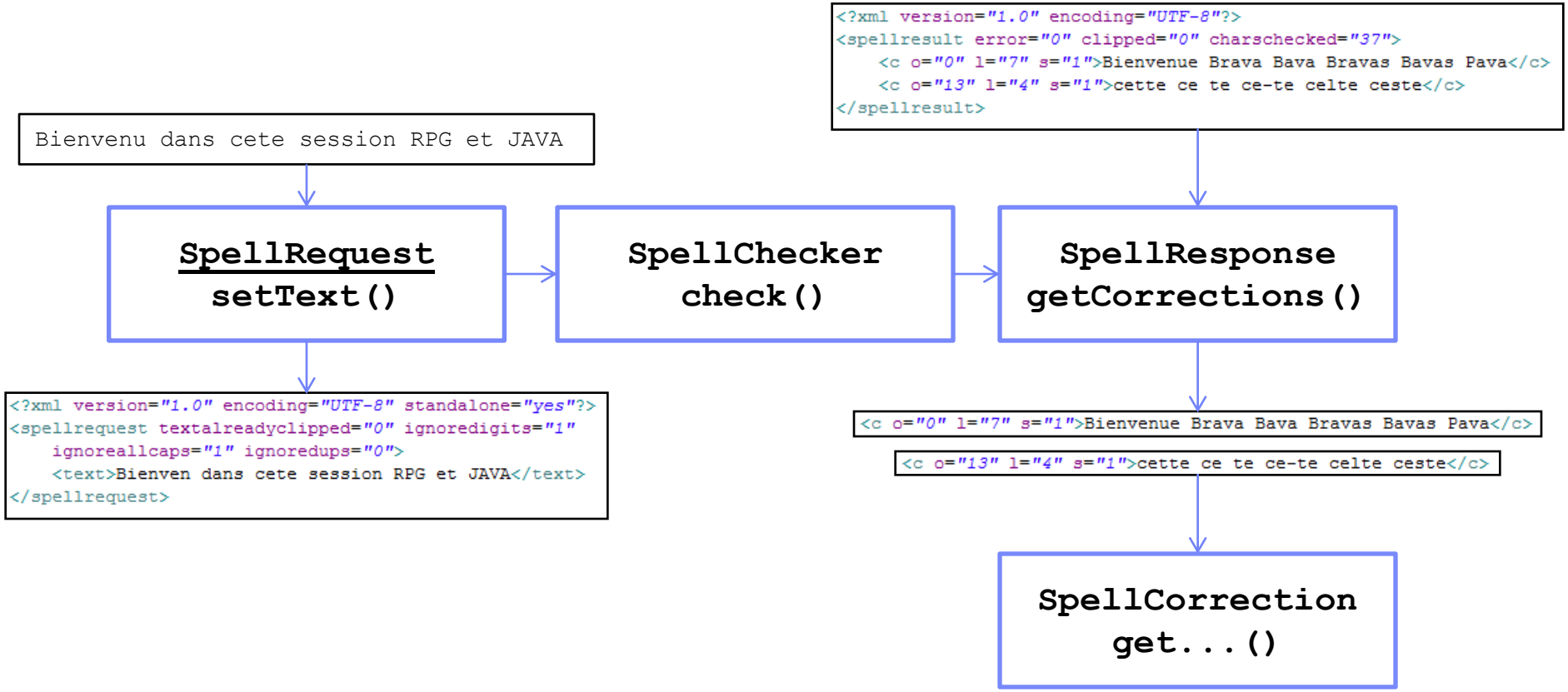
- L'ensemble des classes fournies par Google (google-api-spelling-java-1.1.jar) permettent
  - D'envoyer une requête http/https

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<spellrequest textalreadyclipped="0" ignoredigits="1"
    ignoreallcaps="1" ignoredups="0">
    <text>Bienven dans cete session RPG et JAVA</text>
</spellrequest>
```

  - De décoder la réponse

```
<?xml version="1.0" encoding="UTF-8"?>
<spellresult error="0" clipped="0" charschecked="37">
    <c o="0" l="7" s="1">Bienvenue Brava Bava Bravas Bavas Pava</c>
    <c o="13" l="4" s="1">cette ce te ce-te celte ceste</c>
</spellresult>
```

Fonctionnement : principales classes utilisées

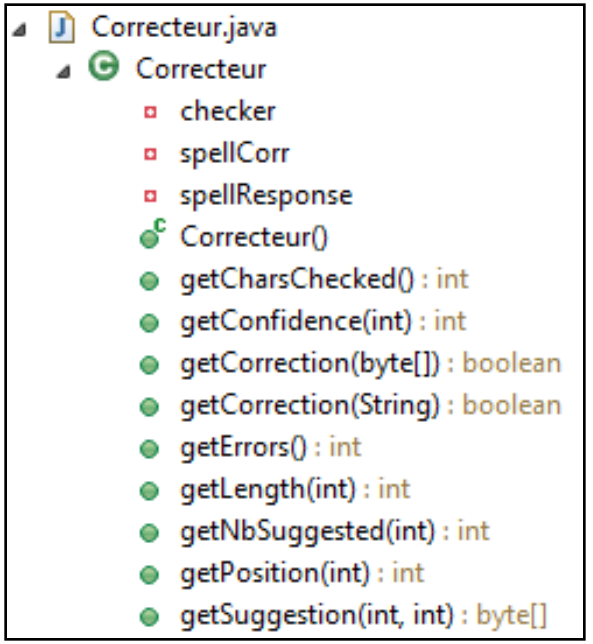


Choix d'implémentation

- La manipulation d'instances JAVA dans le RPG nécessite un travail de prototypage important et méticuleux
  - Il est souvent préférable de créer une enveloppe JAVA
    - Des méthodes prenant des paramètres de type natif
      - int, byte, boolean … au lieu de String …
    - Un nombre minimum de méthodes qui encapsulent toute la mécanique JAVA sous-jacente
      - Instanciations, initialisations, conversion vers et depuis les types natifs
      - Gestion des exceptions pour communiquer un code retour vers le RPG
- En résumé
  - Un code JAVA simple et technique
  - Qui permet de simplifier le code RPG

Classe Correcteur

- Méthodes
  - `boolean getCorrection(byte[] toCheck)`
    - Appel au correcteur
  - `Correcteur()`
    - Constructeur
  - `int getErrors()`
    - Nombre d'erreurs trouvés
  - `int getLength(int n)`
    - Longueur des caractères pour l'erreur `n`
  - `int getConfidence(int n)`
    - Indice de confiance pour l'erreur `n`
  - `byte[] getSuggestion(int n, int taille)`
    - Liste des suggestions pour l'erreur `n`
    - Renvoie une « zone alpha » contenant l'ensemble des suggestions. Chaque suggestion a la taille `taille`



Classe Correcteur

```
import org.xeustechnologies.googleapi.spelling.*;

public class Correcteur {

    // instance du correcteur google
    private SpellChecker checker;
    // instance de la réponse reçue
    private SpellResponse spellResponse;
    // liste des corrections
    private SpellCorrection[] spellCorr;


    // Constructeur par défaut
    public Correcteur() {
        super();

        // Indication du proxy
        Configuration config = new Configuration();
        config.setProxy("proxy", 8080, "http");
        // instanciation du correcteur google
        checker = new SpellChecker(config);
        checker.setOverHttps(true);
        checker.setLanguage(Language.FRENCH);
    }
```

```
/**
 * @return Nombre de caractères contrôlés
 */
public int getCharsChecked() {
    return (spellResponse == null) ? 0 : spellResponse.getCharsChecked();
}

/**
 * @param Rang de la correction
 * @return Nombre de suggestions pour la correction n
 */
public int getNbSuggested(int n) {
    // aucune correction de niveau n
    if (n < 0 || n > spellCorr.length)
        return 0;
    return (spellCorr.length < n) ? 0 : spellCorr[n - 1].getWords().length;

}
```

Classe Correcteur

```
/**
 * @param toCheck : Chaine de caractères à valider
 * @return true si la correction s'est bien déroulée, false sinon
 * @doc Effectue l'appel pour la demande de correction. Stocke le résultat
 */
public boolean getCorrection(String toCheck) {
    // Instanciation de la requête
    SpellRequest request = new SpellRequest();
    // valorisation
    request.setText(toCheck.trim());
    // interrogation
    spellResponse = checker.check(request);
    // test du résultat
    if (spellResponse == null)
        return false;
    // extraction de la liste des corrections
    spellCorr = spellResponse.getCorrections();
    // fin OK
    return true;
}

/**
 * @param toCheck : Chaine de caractères à valider
 * @return true si la correction s'est bien déroulée, false sinon
 * @doc Effectue l'appel pour la demande de correction. Stocke le résultat
 */
public boolean getCorrection(byte[] toCheck) {
    // wrapper
    return getCorrection(new String(toCheck));
}
```

Exemple d'utilisation en JAVA

```
// Instanciation
Correcteur corr = new Correcteur();

// Appel
if (!corr.getCorrection("Et de deu boullettes !")) {
    System.out.println("Appel en ERREUR. Sortie immédiate");
    System.exit(0);
}

// Retrouver les informations générales
System.out
        .println("Nombre de corrections trouvées -> getErrors()       : "
                + corr.getErrors());
System.out
        .println("Nombre de caractères contrôlés -> getCharsChecked() : "
                + corr.getCharsChecked());

// Retrouver le détail de chaque correction
for (int i = 0; i < corr.getErrors(); i++) {
    System.out.println("Correction n° " + (i + 1));
    System.out
            .println("..Longueur chaîne d'origine  -> getLength(i)      : "
                    + corr.getLength(i + 1));
    System.out
            .println("..Position chaîne d'origine  -> getPosition(i)    : "
                    + corr.getPosition(i + 1));
    System.out
            .println("..Nombre de suggestions      -> getNbSuggested(i) : "
                    + corr.getNbSuggested(i + 1));
    System.out
            .println("..Indice de confiance (1ère) -> getConfidence(i)  : "
                    + corr.getConfidence(i + 1));
    System.out
            .println("..Liste des suggestions      -> getSuggestion(i, 20) : "
                    + new String(corr.getSuggestion(i + 1, 20)));
}
```

Résultat

E	t		d	e		d	e	u		b	o	u	l	l	e	t	t	e	s		!
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

```
Nombre de corrections trouvées -> getErrors()       : 2
Nombre de caractères contrôlés -> getCharsChecked() : 22
Correction n° 1
..Longueur chaîne d'origine  -> getLength(i)      : 3
..Position chaîne d'origine  -> getPosition(i)    : 7
..Nombre de suggestions      -> getNbSuggested(i) : 6
..Indice de confiance (1ère) -> getConfidence(i)  : 1
..Liste des suggestions      -> getSuggestion(i, 20) : feu                  deux                 due                  de
Correction n° 2
..Longueur chaîne d'origine  -> getLength(i)      : 10
..Position chaîne d'origine  -> getPosition(i)    : 11
..Nombre de suggestions      -> getNbSuggested(i) : 5
..Indice de confiance (1ère) -> getConfidence(i)  : 1
..Liste des suggestions      -> getSuggestion(i, 20) : boulettes            boulette             bouletés             poulettes
```

En RPG

```
h thread( *serialize )


 // Prototypes pour la classe Correcteur
 // ------------------------------------

 // Constante nom de la classe
d class_Corr      c                   const('utils.correcteur.Correcteur')

 // Constructeur
d corr_new        pr                  o   extproc(  *java
d                                             : class_Corr
d                                             : *constructor )
d                                         class( *java : class_Corr )

 // boolean getCorrection(byte[] toCheck)
d corr_getCorrection...
d                 pr                  n   extproc( *java
d                                             : class_Corr
d                                             : 'getCorrection' )
d  toCheck                    65535a    const options( *varsize )
```

60 lignes de déclarations

Package + classe

Préfixe = corr_

Byte[] ⇔ A

- `thread(*serialize)`
  - Permet au RPG de s'exécuter en toute sécurité dans un environnement multi-threadé
  - Autre valeur `*concurrent`
  - Cela concerne surtout le stockage statique (au niveau du module)

RPG - suite

```
d obj_corr        s                 o   class( *java : class_Corr )
d msg             s              52a   inz
d i               s              10i 0 inz
d nb_err          s              10i 0 inz

 /free

  // Instanciation
  obj_corr = corr_new() ;

  // Appel
  if ( not corr_getCorrection( obj_corr
                               :'Et de deu boullettes !' ) ) ;
     dsply 'Appel en ERREUR. Sortie immédiate' ;
     return ;
  endif ;

  // Retrouver les informations générales
  nb_err = corr_getErrors( obj_corr ) ;
  dsply 'Nombre de corrections trouvées -> getErrors()' ;
  dsply ( '..' + %char( nb_err ) ) ;
  dsply 'Nombre de caractères contrôlés -> getCharsChecked()' ;
  dsply ( '..' + %char( corr_getCharsChecked( obj_corr ) ) ) ;
```

Instance de Correcteur

Instanciation (appel au constructeur)

Appel au correcteur
le 1er paramètre est l'instance

Retrouver les valeurs en retour

RPG - Suite

```
// Retrouver le détail de chaque correction
for i = 1 to nb_err ;
   dsply ( 'Correction n° ' + %char( i ) ) ;
   dsply ( '..Longueur chaîne d''origine  -> getLength(i)' ) ;
   dsply ( '      ' + %char( corr_getLength( obj_corr : i ) ) ) ;
   dsply ( '..Position chaîne d''origine  -> getPosition(i)' ) ;
   dsply ( '      ' + %char( corr_getPosition( obj_corr : i ) ) ) ;
   dsply ( '..Nombre de suggestions       -> getNbSuggested(i)' ) ;
   dsply ( '      ' + %char( corr_getNbSuggested( obj_corr : i ) ) ) ;
   dsply ( '..Indice de confiance (1ère) -> getConfidence(i)' ) ;
   dsply ( '      ' + %char( corr_getConfidence( obj_corr : i ) ) ) ;
   dsply ( '..Liste des suggestions       -> getSuggestion(i, 20)' ) ;
   msg = '      ' + corr_getSuggestion( obj_corr : i : 20 ) ;
   dsply msg ;
endfor ;
```

```
Nombre de corrections trouvées -> getErrors()                2
Nombre de caractères contrôlés -> getCharsChecked()          22
Correction n° 1
..Longueur chaîne d'origine  -> getLength(i)                 3
..Position chaîne d'origine  -> getPosition(i)               7
..Nombre de suggestions      -> getNbSuggested(i)            6
..Indice de confiance (1ère) -> getConfidence(i)             1
..Liste des suggestions      -> getSuggestion(i, 20)
          feu                  deux                  due
Correction n° 2
..Longueur chaîne d'origine  -> getLength(i)                 10
..Position chaîne d'origine  -> getPosition(i)               11
..Nombre de suggestions      -> getNbSuggested(i)            5
..Indice de confiance (1ère) -> getConfidence(i)             1
..Liste des suggestions      -> getSuggestion(i, 20)
          boulettes            boulette              bouleté
```

JVM

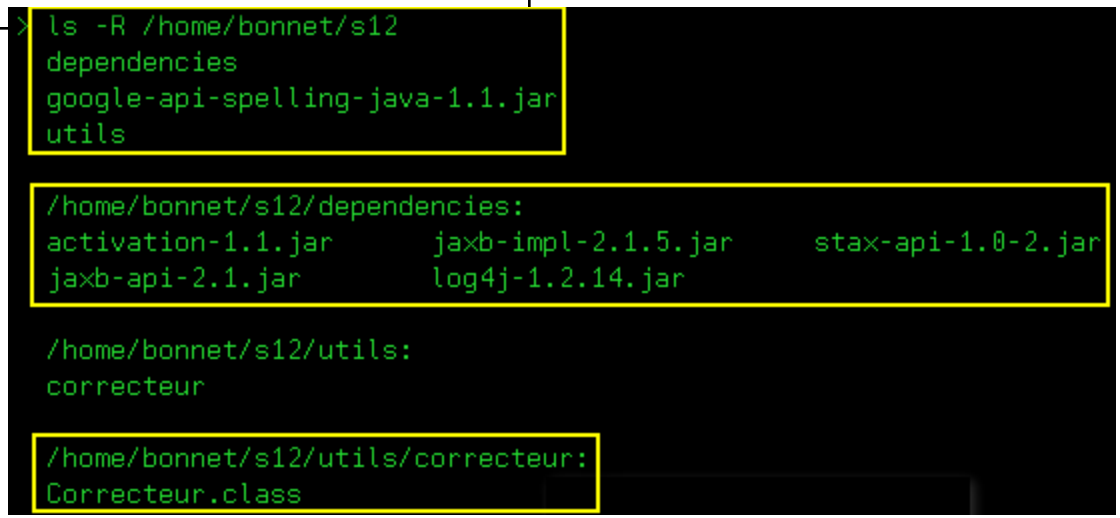
- La JVM est démarrée par un travail, interactif ou batch
  - Lors du 1er appel JAVA
  - De façon explicite (API)
- Elle reste alors liée à ce travail
  - Seul l'arrêt du travail permet d'arrêter la JVM
- Un travail ne peut avoir qu'une unique JVM instanciée
  - Il est donc important de bien contrôler ses propriétés de démarrage
    - Version de la JVM
    - Autres propriétés (à suivre)

JVM

- Contrôle de la JVM
  - Par des variables d'environnement
    - JAVA_HOME
      - Le répertoire IFS de la JVM à invoquer
    - CLASSPATH
      - Une liste d'exécutables Java (fichier .jar) ou de répertoires (contenant des classes)
      - Rappel : un .jar contient des fichiers .class qui sont les « exécutables » Java pour la JVM
  - Ces deux propriétés sont les plus importantes
- Pour voir les travaux avec une JVM instanciée
  - WRKJVMJOB

JVM

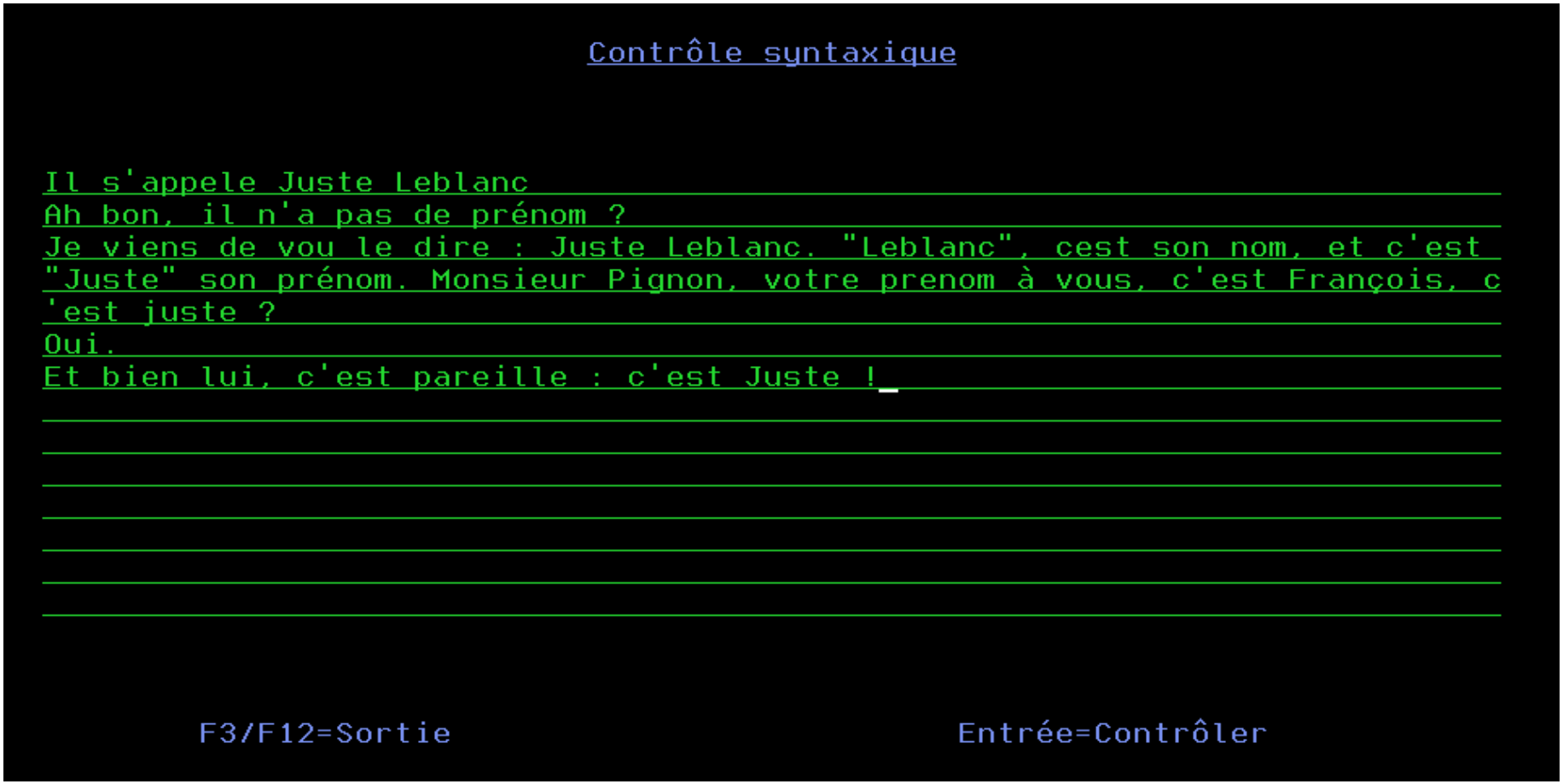
```
pgm

/* JVM 6.0 */
ADDENVVAR  ENVVAR(JAVA_HOME) +
      VALUE('/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit') LEVEL(*JOB) +
      REPLACE(*YES)

/* CLASSPATH */
ADDENVVAR  ENVVAR(CLASSPATH) +
      VALUE('.:/home/bonnet/s12/google-api-spelling-java-1.1.jar:/home/bonn+
      et/s12/dependencies/activation-1.1.jar:/home/bonnet/s12/dependencies/+
      jaxb-api-2.1.jar:/home/bonnet/s12/dependencies/jaxb-impl-2.1.5.jar:/h+
      ome/bonnet/s12/dependencies/log4j-1.2.14.jar:/home/bonnet/s12/depende+
      ncies/stax-api-1.0-2.jar') LEVEL(*JOB) REPLACE(*YES)

/* Répertoire courant : contient la classe Correcteur */
cd         '/home/bonnet/s12'

endpgm
```



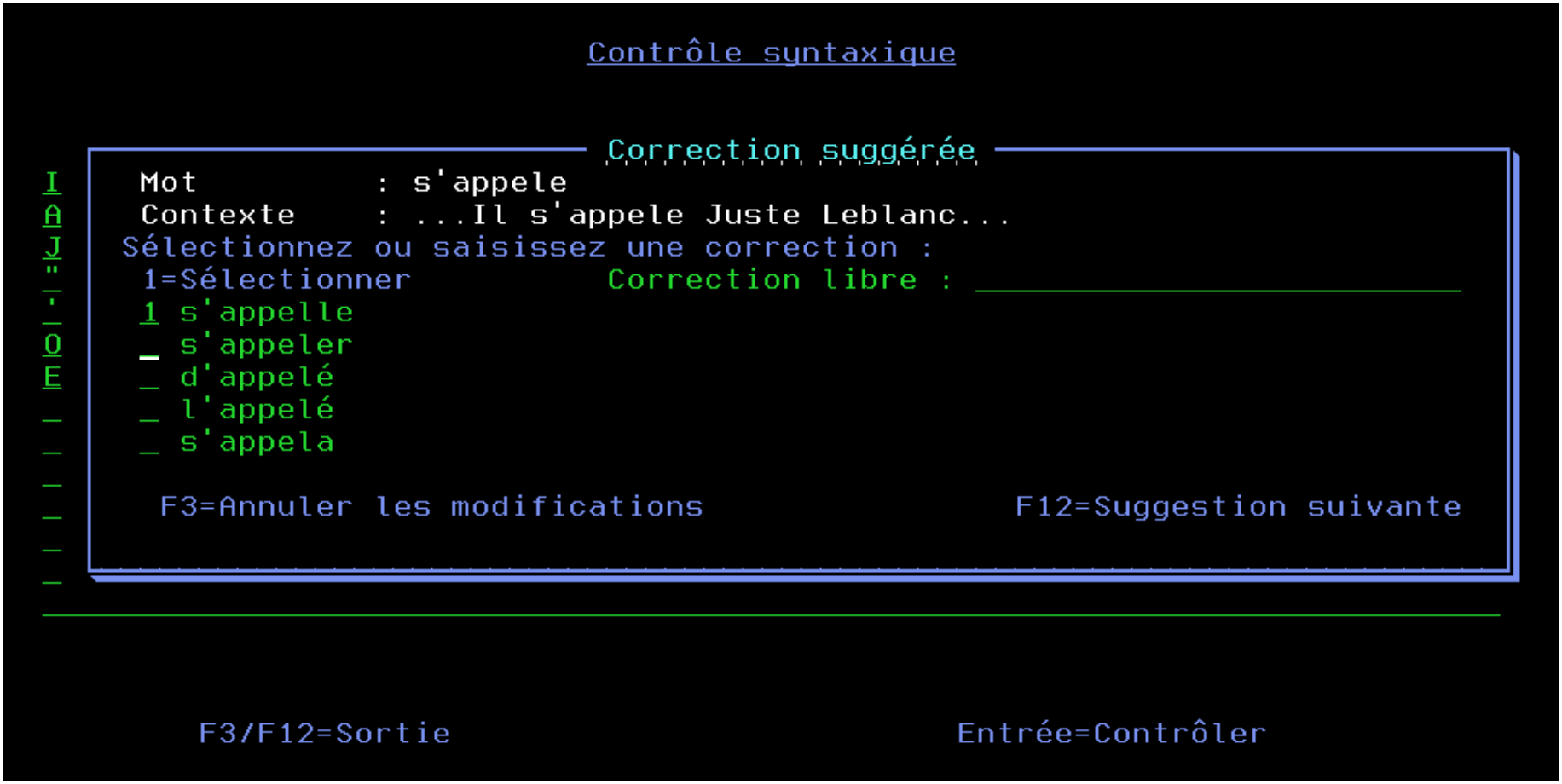
Exemple final

- Nous créons un programme interactif
  - Qui prend en entrée un texte à corriger
  - Affiche un fenêtre avec un sous-fichier pour chaque correction à traiter
  - Prend en compte les corrections
  - Retourne le texte modifié

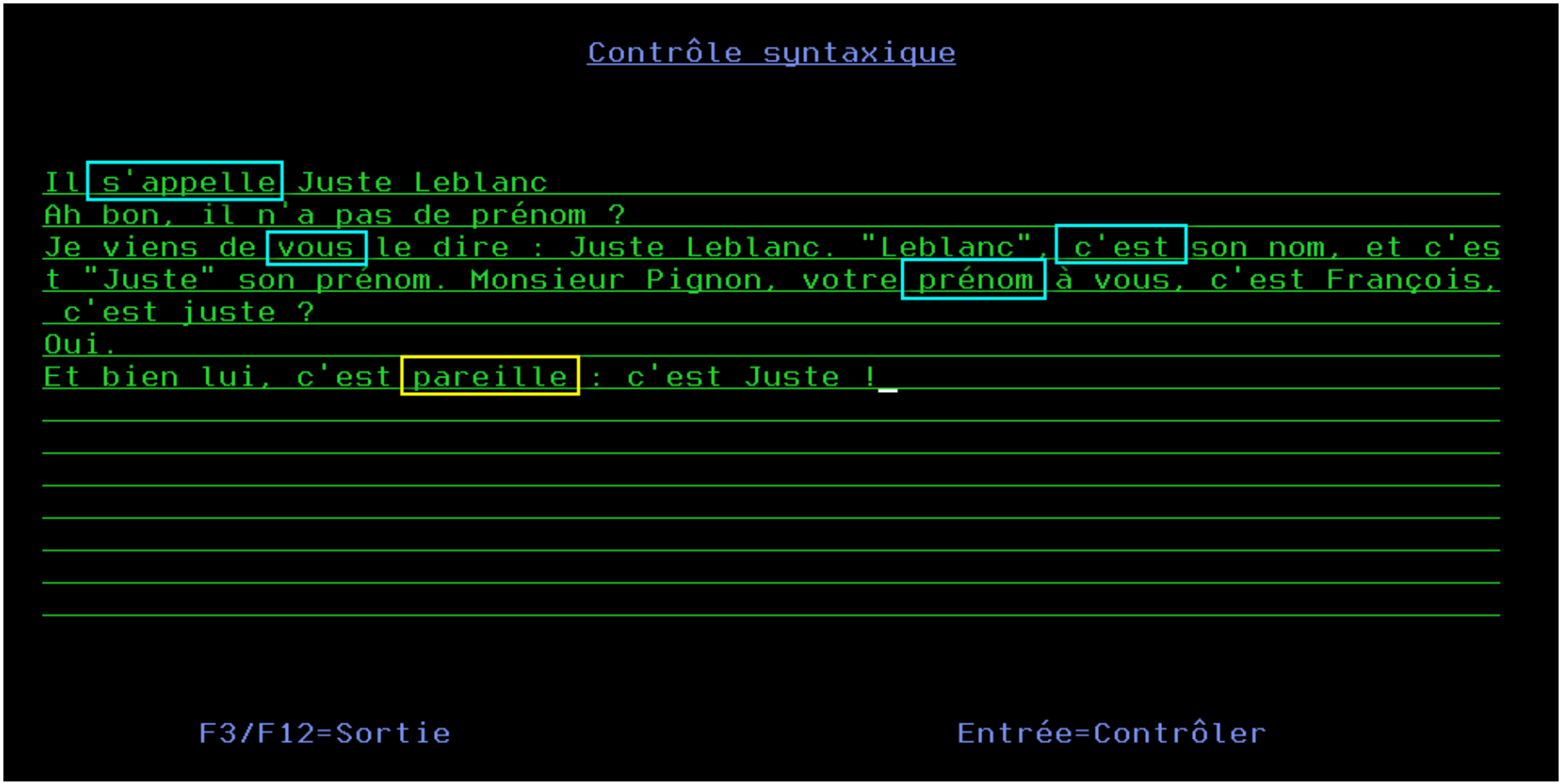
Exemple final



Exemple final



Exemple final



Conclusion

- Code
  - 150 lignes de JAVA (commentaires inclus)
  - 100 lignes de RPG spécifiques aux appels JAVA
  - 1 programme interactif sous-fichier

- Et tous mes programmes 5250 peuvent désormais bénéficier d'un correcteur orthographique
  - Gratuit
  - En ligne (pas de maintenance, mis à jour)

- Et d'autres fonctionnalités possibles
  - Traduction, conversions de devises, météo, cours de la bourse …

Pour aller plus loin

- Contrôle de la JVM
  - JAVA_HOME
    - Détermine la JVM
      - Version de JAVA, mais aussi 32/64 bits
  - CLASSPATH
    - Chemins de recherche des exécutables JAVA
  - QIBM_RPG_JAVA_PROPERTIES
    - Pour les propriétés JAVA
      - Contient les noms et valeurs des propriétés JAVA à transmettre lors du démarrage de la JVM
      - Par exemple, pour router la console et les erreurs dans des fichiers
        - -Dos400.stderr=file:stderr.txt
        - -Dos400.stdout=file:stdout.txt
  - QIBM_RPG_JAVA_EXCP_TRACE
    - Y : génère une trace (fichier) en cas d'exception
    - Dynamique

Pour aller plus loin

- Contrôle de PASE
  - QIBM_USE_DESCRIPTOR
    - Y : indique à PASE que l'on souhaite utiliser les descripteurs standards stdin, stdout et stderr
    - Couplé avec QIBM_RPG_JAVA_PROPERTIES
    - Attention, c'est au programmeur de s'assurer que les fichiers sont ouverts, avec les bons descripteurs

Pour aller plus loin

- On ne peut pas tout faire via RPG
  - Accès aux propriétés publiques impossibles
    - Ce n'est pas un problème en général, le mécanisme d'encapsulation permet d'avoir une méthode équivalente
    - Possible via l'utilisation de JNI
      - Complexe
  - Quelques méthodes ne fonctionnent pas via RPG
    - java.lang.Class
      - forName()
        - Permet de charger dynamiquement une classe
  - Le « garbage Collector » est limité
    - Mais on peut l'aider !

Pour aller plus loin

- JNI (JAVA Native Interface)
  - Fait partie de JAVA
  - Ensemble d'API (natives ou JAVA)
  - Permettant à RPG d'appeler du code JAVA
    - Sous-jacent à tout ce que nous avons vu
  - Permettant à JAVA
    - D'appeler du code RPG
    - D'implémenter certaine méthode en RPG
- L'infocentre fournit du code à utiliser pour aider RPG à piloter le Garbage Collector
  - http://pic.dhe.ibm.com/infocenter/iseries/v6r1m0/index.jsp?topic=/rzasc/sc092507252.htm
  - Permet d'indiquer quelles sont les instances à conserver ou à libérer

Gestion des exceptions

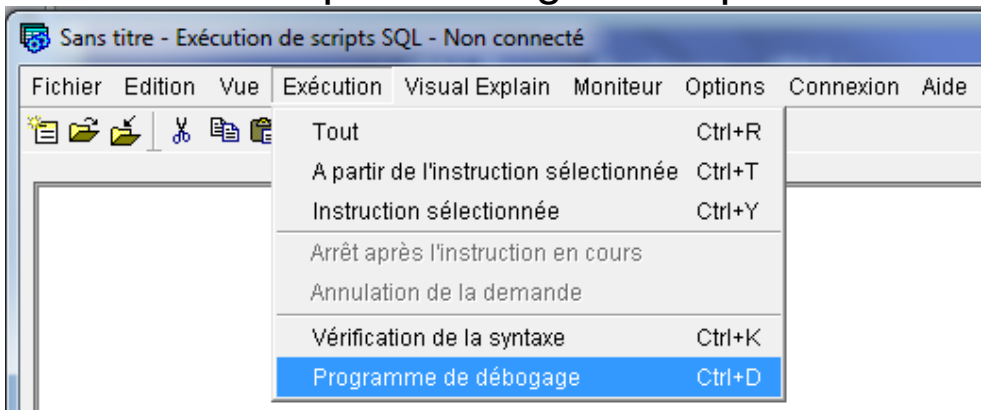
- RPG reçoit un RNX0301
  - Code statut 301
  - Un bloc MONITOR
  - Problème : à la réception du message, on est remonté dans la pile d'appel jusqu'au RPG
- Pas de solution miracle
  - On atteint la limite d'intégration des langages
- Conseil
  - Gérer au maximum les exceptions dans le JAVA
  - Remonter des codes retour au RPG

Mise au point

- Le débogage ne fonctionne qu'avec des .class
  - Ne fonctionne pas avec des .jar …
  - Les fichiers .java (sources) doivent être dans le même répertoire que les .class correspondant
- STRDBG
  - Nécessite deux sessions distinctes
    - 1ère session : exécution
    - 2nde session : STRSRVJOB sur la 1ère
  - Remarques
    - JAVA n'est déboguable qu'après instanciation (appel au constructeur)
      - Le constructeur n'est donc pas déboguable
    - Les variables dans le code JAVA sont affichées, mais non modifiables

Mise au point

- System Debugger
  - Outils JAVA disponible
    - Sur l'IBM i : /QIBM/ProdData/HTTP/Public/jt400/lib
    - A copier sur son poste local
    - A lancer par
      - `java utilities.DebugMgr`
      - `java utilities.Debug`
  - Par l'exécution de script SQL
    - Le même outil peut déboguer les procédures cataloguées



  - Permet de modifier les variables JAVA

Mise au point

- RDP
  - Rational Developer for Power
  - Recommandé
    - Pour le développement : un seul outil pour développer RPG et JAVA
    - Pour la mise au point : un seul outil pour déboguer RPG et JAVA
  - Permet de modifier les variables JAVA
  - Les points d'entrée de service sont la méthode la plus souple pour déboguer

Mise au point

- En phase de développement
  - Côté JAVA, routez la sortie standard vers un fichier
  - Faites de print : System.out.println
  - Cela reste la méthode la plus simple
  - D'autres possibilités sont efficaces
    - log4j
      - Outil de log paramétrable
      - Permet d'avoir le même code en développement et production, d'activer les traces en production si nécessaire …

Nous contacter

- Par mail
  - nbonnet@gaia.fr
  - contact@gaia.fr
- Nos sites
  - www.gaia.fr
  - www.know400.fr
  - www.as400.fr