Détection et traitements sur les noms d'objet (et autres) dans les textes.

Détection et traitements sur les noms d'objet (et autres) dans les textes.

List of Figures

List of Tables

DJIN (Detection in Journals of Identifiers and Names)

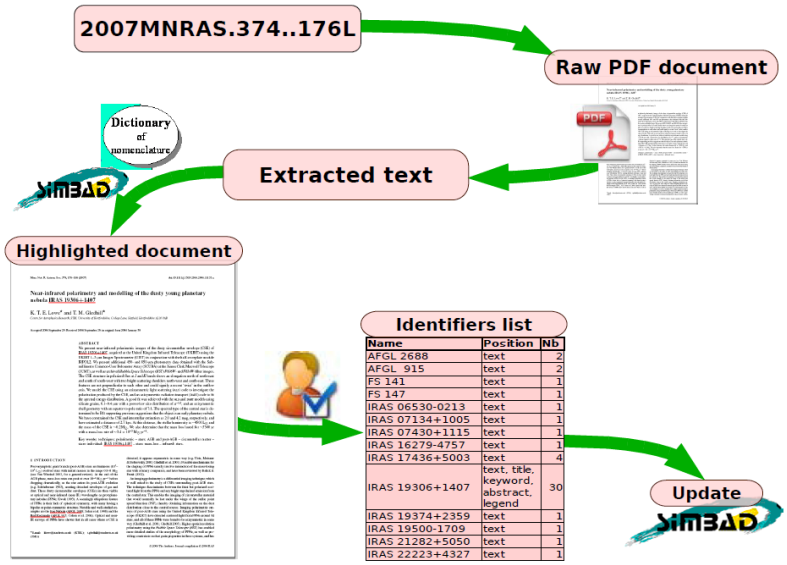
1. Généralités

1.1. Objectif

Le but de DJIN est de :

- reconnaître automatiquement les noms d'objet dans un article au format PDF
- les mettre en valeur dans le document original sans changer la présentation
- en déduire une liste d'identificateurs à ajouter dans **SIMBAD** pour ce bibcode
- générer le fichier de commandes de mises à jour et l'exécuter dans **SIMBAD**
- les noms d'objets peuvent également être détectés et mis en valeur dans des documents HTML au cas où la version PDF ne serait pas disponible ou exploitable

Figure 1. Schéma du fonctionnement général de DJIN



1.2. Par où commencer ?

Dans l'écran principal, la partie droite affiche le texte extrait du document traité et la partie gauche une arborescence des noms d'objet détectés regroupés par ordre alphabétique.

1. Lancer le programme en exécutant le fichier `objname.bat` sous **Windows** ou `objname.sh` sous **Unix** depuis le répertoire de DJIN.

2. Dans la zone de saisie du bibcode entrer le code bibliographique de l'article désiré et dans le menu *File*, lancer la commande *Open PDF from Bibcode*.

   Le programme recherche alors l'emplacement de l'article, télécharge le fichier, extrait le texte du document et lance la recherche des noms d'objet.

   Si le contenu de l'article n'a pas pu être localisé, une page Web est affichée et l'utilisateur doit alors cliquer sur le lien qui permet d'accéder au contenu PDF du document.

Pour utiliser un contenu HTML au lieu de documents PDF, lancer la commande *Open HTML from bibcode* du menu *File*. Les noms d'objet sont affichés dans l'arborescence de gauche avec leur nombre d'occurrences et l'endroit du document où ils ont été trouvés (title, abstract, keyword, subtitle, text, legend).

3. Dans le menu *Name* lancer la commande *Check found names* pour vérifier que les noms existent bien dans **SIMBAD** (ou cocher la case *Automatically check names in SIMBAD* dans les préférences.)

   Les noms existant apparaîssent en vert s'ils sont déjà sur la référence et en rouge sinon. Les noms d'objet inexistants sont en bleu si leur format existe ou en violet sinon.

4. Dans le menu *File* lancer la commande *View annotated document* pour visualiser dans **Acrobat Reader** le document PDF dans lequel les noms d'objet sont soulignés.

   Les noms sont soulignés en vert, rouge, bleu ou violet selon leur existence dans **SIMBAD** et leur présence sur la référence.
   En cliquant sur les noms soulignés, on peut voir leur description dans **SIMBAD**.

5. Pour remplir la liste des identificateurs à ajouter à la référence, les noms d'objet peuvent être ajoutés

   - un par un avec la commande *Identifier, Add a name...*
   - tous ensemble avec la commande *Identifier, Add all names...*
   - tous les noms sélectionnés avec la commande *Identifier, Add selected names...*
   - depuis le presse-papier avec la commande *Identifier, Add from clipboard...* Des colonnes entières de noms d'objet peuvent ainsi être sélectionnées et copiées depuis **Acrobat Reader**.

6. La liste des identificateurs apparaît automatiquement après un ajout multiple de noms d'objet. Sinon elle peut être appelée par la commande *Identifier, Manage identifiers...* (un bibcode valide doit avoir été entré).

7. Le script de mise à jour peut être testé avec le bouton *Simulate* ou être définitivement envoyé avec le bouton *Execute...*.

Les commandes qui ont une icône dans le menu peuvent également être lancées depuis le bouton de la barre d'outil qui présente la même icône.

2. Fonctionnalités

2.1. Fichiers

2.1.1. Ouverture d'article à partir d'un bibcode

A partir du bibcode, DJIN se connecte à un serveur bibliographique pour localiser l'URL du contenu de l'article.

La classe `ActionOpenBibcode` est invoquée lors de l'ouverture d'un article PDF (menu *File, Open PDF from bibcode* ou bouton ) ou d'un article HTML (menu *File, Open HTML from bibcode* ou bouton ) depuis un bibcode.

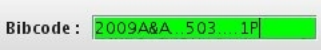
Dans les deux cas, il y a d'abord validation du bibcode, puis détermination de l'URL du contenu PDF ou HTML, et enfin ouverture du document.

L'ouverture d'un PDF depuis le bibcode suivant (*File, Open PDF from next bibcode*) ajoute simplement le nombre de pages du document PDF courant au numéro de page du bibcode et lance l'ouverture de l'article à partir du nouveau bibcode.

2.1.2. Complétion du bibcode

Figure 2. Zone d'édition du bibcode

Une requête est envoyée à SIMBAD pour vérifier et éventuellement compléter le code bibliographique (avec le bon nombre de “.” et la bonne lettre d'auteur.



La vérification/complétion du bibcode est lancée lorsque l'utilisateur effectue l'une des actions suivantes :

- appui sur Entrée dans la zone de saisie du bibcode
- ouverture de document PDF depuis un bibcode
- ouverture de document HTML depuis un bibcode
- Affichage de la liste des identificateurs
- Après l'ajout de plusieurs identificateurs dans cette liste

La zone d'édition s'affiche en vert si le bibcode existe dans SIMBAD et en rouge sinon.

Lorsque la vérification a été faite, elle n'est plus retentée à moins que le contenu de la zone ne change.

Une requête est envoyée à l'adresse contenue dans le paramètre urlbibcode (c'est l'adresse d' exécution des scripts par SIMBAD) à laquelle est ajoutée « query bibcode » et le bibcode. Le résultat est parcouru à la recherche de lignes commençant par un bibcode. S'il n'y en a aucune, la zone du bibcode s'affiche en rouge. S'il y en a une elle s'affiche en vert et présente le bibcode complet. S'il y a plusieurs (par exemple avec « 2009a&a.503.1 »), une liste déroulante est proposée pour choisir parmi les possibilités.

Si un bibcode valide est trouvé :

- Le nom du journal est sélectionné dans la liste des journaux.
- Son numéro de volume s'affiche
- La liste simbo est récupérée en utilisant le web service `queryObjectByBib` (elle servira à afficher les noms d'objet déjà présents en vert).
- Le nombre de noms d'objet sur la référence s'affiche dans la barre d'état

Les traitements sur les bibcodes sont dans la classe BibcodeListener

2.1.3. URL du contenu PDF d'un article

Cette URL est déterminée lorsque l'ouverture d'un document PDF est demandée à partir du bibcode ou lors du traitement d'une liste de bibcode si les chemins vers les articles ne sont pas précisés.

Une première URL est obtenue en ajoutant le bibcode au contenu du paramètre urldownload.

Si le paramètre urlcommand est renseigné la commande qu'il contient (**wget %s -O**) est exécutée pour récupérer le fichier PDF. L'URL vient remplacer la chaîne **%s** (utilisation de `java.lang.String.format`) et le chemin d'un fichier temporaire est ajouté à la fin. La sortie

standard de la commande est affichée dans la fenêtre principale. Le fichier temporaire est alors censé contenir le PDF de l'article.

Si le paramètre urlcommand est vide, l'URL obtenue est utilisée pour ouvrir une connection et télécharger le contenu.

Le début du contenu téléchargé est analysé pour voir s'il commence par "%PDF-" (entête des documents PDF).

Si oui, l'extraction de texte commence.

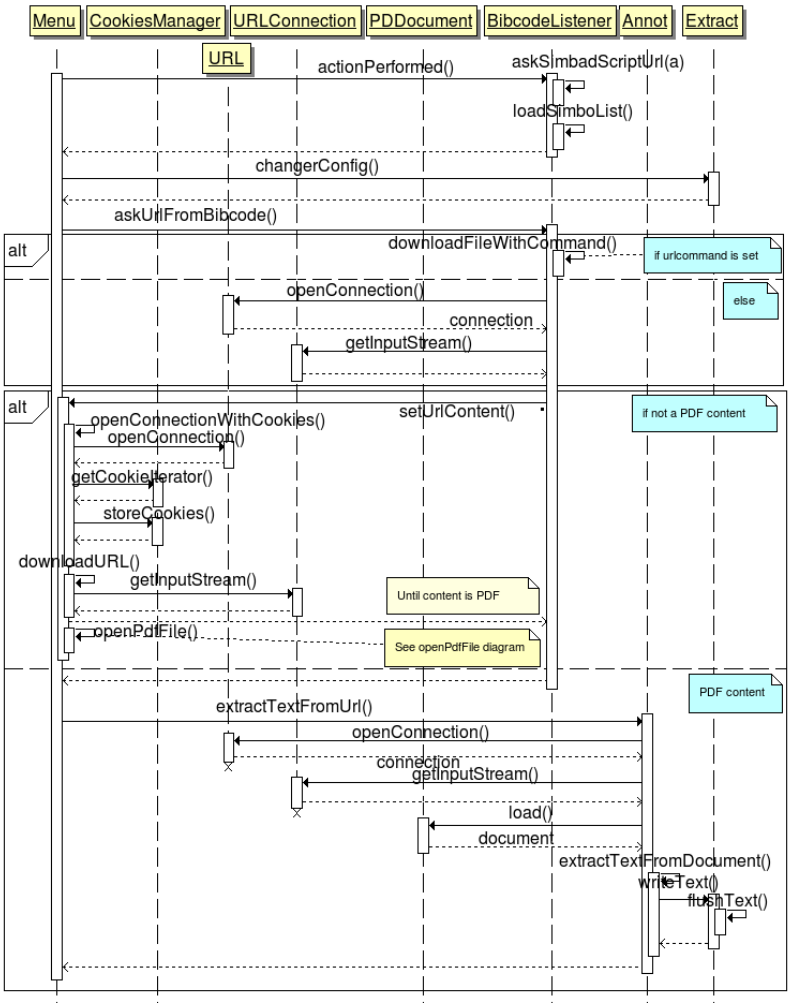
Sinon, le contenu HTML du fichier est affiché dans la fenêtre principale (DJIN se comporte alors comme un navigateur) et l'utilisateur peut cliquer sur les liens pour trouver contenu de l'article.

A chaque clic de l'utilisateur, la fonction `HyperlinkEvent.hyperlinkUpdate` associée à l'objet `JEditorPane` qui affiche le contenu appelle la fonction `Menu.setUrlContent` qui télécharge le nouveau contenu et vérifie si c'est un fichier PDF.

Cookies : à chaque ouverture de connexion URL, le programme teste si un fichier du répertoire `cookie` correspond au nom de domaine de l'URL. Si oui, les cookies du fichier (une ligne=un cookie) sont intégrés à la requête (fonction `CookiesManager.getCookieIterator`).

De même, si des cookies sont présents dans l'entête de la connexion, ils sont enregistrés dans un fichier de ce répertoire par la fonction `CookiesManager.storeCookies`.

Figure 3. Diagramme de la validation d'un bibcode puis détermination de l'URL du contenu



2.1.4. Ouverture de documents PDF

Cette ouverture consiste à charger le document en mémoire grâce à la librairie PDFBox, à en extraire le texte, puis à y détecter les noms d'objet.

La lecture du contenu d'un document PDF peut se faire en ouvrant directement le fichier ou en le désignant par son bibcode.

La fonction `Annot.extractTextFromDocument` est appelée suite aux commandes suivantes du menu *File : Open PDF file, Open PDF from URL, Open PDF from bibcode, Open PDF from next bibcode*.

Que ce soit depuis un fichier ou une URL, le contenu complet est (télé)chargé par la fonction `load` de la librairie PDFBox qui interprète le document PDF.

Le texte est alors extrait par l'objet Extract dérivé de PDFTextStripper de la librairie PDFBox dans lequel la fonction `flushText` a accès aux caractères dans l'ordre de lecture (liste `org.pdfbox.util.PDFTextStripper.charactersByArticle`). Ce qui permet de s'affranchir des problèmes liés aux textes sur plusieurs colonnes. Tous les documents PDF analysés, sauf les NewA, contenaient cette information.

Par contre, les légendes de figure et les tables sont quand même insérées au milieu du texte et il faut les séparer en fonction de la taille de leurs caractères.

C'est dans la fonction `flushText` que sont utilisés la pluspart des paramètres d'extraction, que sont éliminés les entêtes et pied de page, ajoutés les espaces en fonction des distances entre caractères (et les tabulations pour les colonnes de table) et reconnus les symboles graphiques. Les caractères extraits avec leur position sont ajoutés à l'objet `Sortie` qui contient le résultat de l'extraction de texte.

Une fois le texte extrait, la fonction `Sortie.determinerType` affecte un type à chaque ligne qu'elle choisit parmi la liste suivante :

- `ObjLigne.TITLE`
- `ObjLigne.KEYWORD`
- `ObjLigne.ABSTRACT`
- `ObjLigne.SUBTITLE`
- `ObjLigne.TEXT`
- `ObjLigne.LEGEND`
- `ObjLigne.TABLE`
- `ObjLigne.AUTHOR`
- `ObjLigne.REFERENCE`

Puis, la fonction `Sortie.traiterSortie` recolle ensemble les lignes de texte (et leur liste de positions associée) de même type pour éviter que les noms d'objet à cheval sur deux lignes ne soient pas détectés.

La détection des noms d'objet peut commencer.

Figure 4. Diagramme de l'extraction de texte d'un document PDF suivie de la recherche et vérification des noms d'objet

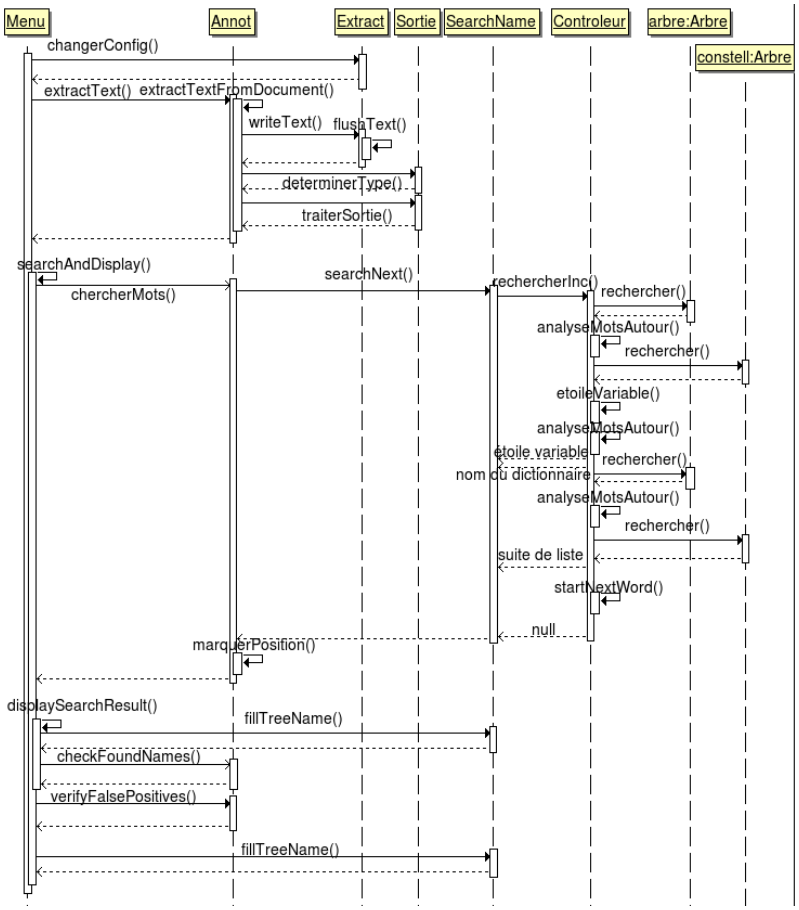
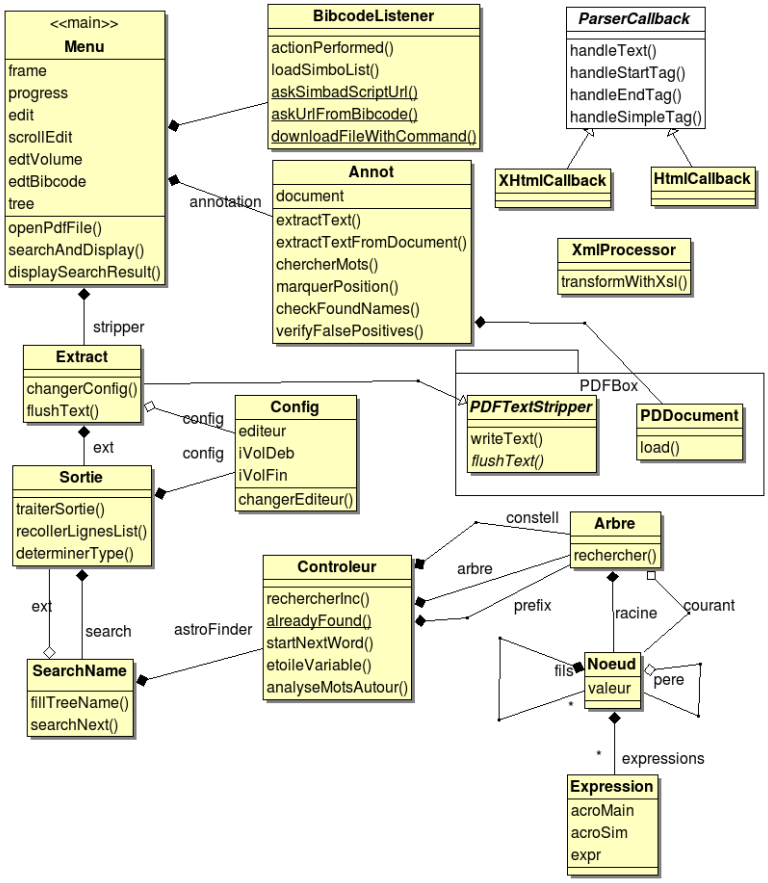


Figure 5. Diagramme des classes impliquées dans le traitement d'un document



2.1.5. URL du contenu HTML d'un article

Pour obtenir l'URL du contenu d'un article, deux stratégies sont possibles selon qu'on peut accéder au contenu directement avec DJIN ou au contraire qu'un navigateur soit nécessaire pour y parvenir. Bien que l'échange de cookies ait été implémenté dans DJIN, certains contenus lui sont toujours inaccessibles (ex. : les MNRAS).

Pour le premier cas, l'URL est obtenue en concaténant le bibcode au paramètre urldownloadhtml, l'utilisateur ne s'occupe de rien.

Pour le deuxième cas, l'URL précédente est ouverte dans un navigateur et l'utilisateur doit naviguer jusqu'à ce qu'il ait le contenu de l'article dans un cadre unique (pas de frames). Il va ensuite dans le menu *Outils*, *Send to DJIN* (cette commande est ajoutée par l'extension Firefox). Ce qui a pour effet de sauvegarder la page courante dans le répertoire temporaire et dans le fichier `firefoxContent.htm` puis d'envoyer un message SAMP à DJIN pour lui dire de traiter ce fichier.

Le traitement des messages SAMP reçus est effectué par la fonction `SampMessageHandler.processCall`.

2.1.6. Ouverture de documents HTML

A partir d'une page contenant l'article, DJIN doit maintenant en extraire le texte en filtrant au passage les éléments inutiles.

La fonction `Annot.readTextFile` peut être appelée lors du traitement d'un document d'une liste, l'ouverture d'article en HTML depuis un bibcode (*File, HTML from bibcode* ou ), l'ouverture d'un fichier texte ou HTML ou la réception d'un message SAMP.

Pour l'ouverture depuis une URL, la connexion est ouverte avec éventuellement échange de cookies comme pour les URL vers des documents PDF.

Ensuite, la difficulté pour DJIN de lire les articles au format HTML est souvent de repérer le contenu de l'article parmi tous les éléments de la page HTML (entêtes, menu, publicités, ...). C'est pourquoi le paramètre filterHtml peut contenir le chemin d'un fichier de transformation XSL qui va permettre d'isoler le contenu de l'article. Des fichier XSL ont déjà été créés pour les jounaux NewA, NewAR, IBVS, MNRAS.

Comme la transformation XSL ne peut fonctionner que sur un fichier XML, le fichier HTML est d'abord mis au format XHTML qui est de l'HTML respectant la norme XML par la fonction `XmlProcessor.transformWithXsl`. Le résultat va dans le fichier temporaire `djinxhtml.xml` .

Le résultat de la transformation va dans le fichier temporaire `djinhtml.htm` .

La récupération du texte contenu dans les documents HTML est assurée par le parser HTML `HtmlCallback` dérivé de `javax.swing.text.html.HTMLEditorKit.ParserCallback`.

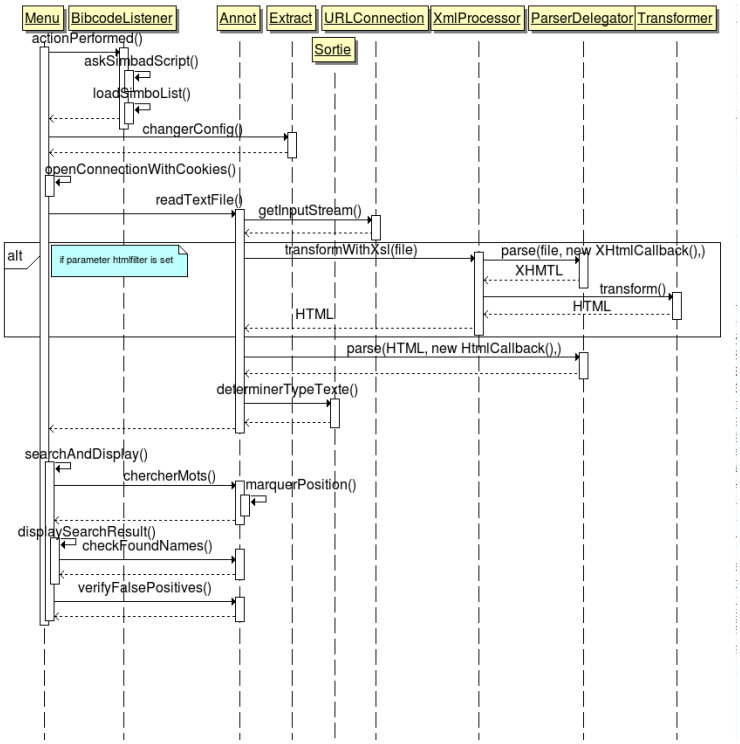
> **Note**
> Ce parser est capable de récupérer les informations de types de positions intégrées au document HTML par l'exportation de texte vers HTML qui est un bon moyen de stocker le résultat de l'extraction du texte d'un PDF.

Le type de chaque ligne est ensuite déterminé par la fonction `Sortie.determinerTypeTexte` parmi les possibilités suivantes :

- `ObjLigne.TITLE`
- `ObjLigne.KEYWORD`
- `ObjLigne.ABSTRACT`
- `ObjLigne.SUBTITLE`
- `ObjLigne.TEXT`
- `ObjLigne.LEGEND`
- `ObjLigne.TABLE`
- `ObjLigne.AUTHOR`
- `ObjLigne.REFERENCE`

Un traitement particulier est réservé aux fichiers ReadMe de VizieR par `Sortie.determinerTypeReadMe`.

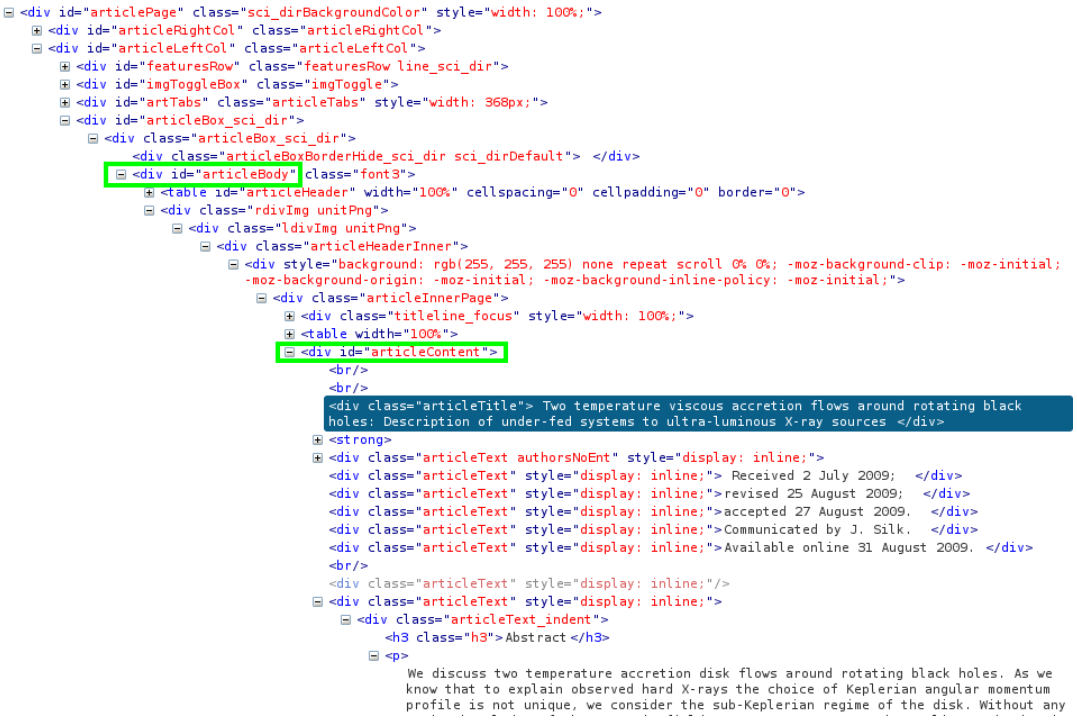
Figure 6. Diagramme de séquence de l'ouverture d'un document HTML à partir du bibcode



Voici un exemple montrant la structure d'un document HTML contenant un article à extraire. Le titre est ici mis en valeur avec la commande *Inspect element* du menu contectuel de Firefox (cette commande fait en fait partie de l'extension Firebug[1] de Firefox). Les éléments récupérés en premier par le fichier XSL sont encadrés en vert.

[1] https://addons.mozilla.org/fr/firefox/addon/1843

Figure 7. Titre d'un article de NewA dans sa page HTML



Voici un extrait du fichier de transformation XSL permettant d'extraire le texte de l'article :

```
<xsl:template match="/">
<html>
  <xsl:apply-templates select="//div[@id='articleBody']//div[@id='articleConten
</html>
</xsl:template>

<xsl:template match="div[@class='articleTitle']">
  <H1><xsl:value-of select="."/></H1>
</xsl:template>

<xsl:template match="div[@class='articleText']">
  <xsl:copy-of select="."/>
</xsl:template>

<xsl:template match="div[@class='graphText']">
 <div class="djinSmall">
  <xsl:copy-of select="."/>
 </div>
</xsl:template>

<xsl:template match="div[@class='refText']">
</xsl:template>
```

Le premier élément `xsl:template` sélectionne tout le contenu du document XHTML (“/”) et lance le traitement par les autres `xsl:template` du contenu des balises `div` avec l'argument `class="articleContent"` qui sont dans une autre balise `div` ayant l'argument `class="articleBody"`. Le résultat de ce traitement est écrit sur la sortie entre deux balises `html`.

La présence des caractères “//” dans le chemin XPATH signifie que d'autres éléments peuvent se trouver entre les balises indiquées.

Le deuxième élément `xsl:template` écrit le titre contenu dans la balise `div` avec argument `class="articleTitle"` située directement dans la balise sélectionnée par le `xsl:template` précédent sur la sortie entre deux balises `H1`.

Le troisième élément `xsl:template` recopie telles quelles toutes les balises trouvées dans les éléments `<div class="articleText">`

Le quatrième élément `xsl:template` recopie telles quelles les balises des éléments `<div class="graphText` dans un élément `<div class="djinSmall">`, ce qui sera interprété par l'extraction de texte depuis l'HTML comme du texte en petit caractères (légende de figure).

Le dernier élément `xsl:template` ne fait rien avec le contenu des éléments `<div class="refText">` qui sera donc ignoré.

2.1.7. Ouverture de documents texte

DJIN peut lire les contenus de type texte depuis un fichier, une URL ou le presse-papier.

C'est aussi la fonction `Annot.readTextFile` qui est appelée pour l'ouverture de texte depuis un fichier ou une URL. `Annot.readTextString` étant appelée pour le presse-papier.

Le type de chaque ligne est ensuite déterminé par `Sortie.determinerTypeTexte`.

2.1.8. Détection et vérification des noms d'objet

Maintenant que le texte a été extrait du document, DJIN recherche les noms d'objet et les vérifie éventuellement dans SIMBAD.

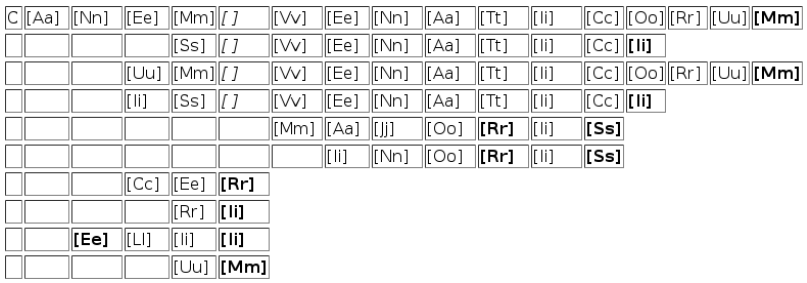
La fonction `Annot.chercherMots` lance la recherche des noms d'objet dans le texte extrait et crée les annotations et marque-pages PDF si un document PDF est chargé (Pour les documents HTML, la mise en valeur et les marque-pages sont générés au moment d'enregistrer le document).

`SearchName.searchNext` recherche la prochaine occurrence d'un nom d'objet et retourne *`true`* tant que la fin du texte extrait n'est pas atteinte. Cette fonction peut gérer la recherche à partir d'une expression régulière ou d'une liste d'expressions ou d'un arbre de recherche (c'est le mode habituel dans DJIN) ou seulement d'un extrait de cet arbre (recherche avec préfixe) selon la façon dont l'objet `SearchName` est initialisé.

Elle appelle pour chaque mot la fonction `Controleur.rechercherInc` qui parcourt l'arbre principal, celui des constellations pour repérer les noms d'étoiles variables et tient compte des mots avant et après et des liste de noms séparés par des virgules ou “and”. La valeur de retour est un objet `ObjName` qui représente un nom d'objet détecté avec les informations sur sa position et les expressions régulières correspondantes.

Un algorithme de recherche en arbre est utilisé dans lequel les expressions régulières sont découpées en éléments correspondant le plus souvent à un caractère. Ces éléments sont regroupés dans un arbre parcouru par l'algorithme pour chaque mot, ce qui permet de n'effectuer qu'une fois les correspondances pour les débuts que les expressions régulières ont en commun. Le tableau suivant représente une partie de l'arbre généré à partir des noms de constellations. Les éléments facultatifs sont en italiques et ceux pouvant terminer un mot cherché (il doivent coïncider avec une fin de mot) sont en gras.

Figure 8. Extrait de l'arbre de recherche issu des noms de constellation



Noms de constellation présents dans cet extrait :

- Canem venaticorum
- Canes venatici
- Canum venaticorum
- Canis venatici
- Canis majoris
- Canis major
- Canis minoris
- Canis minor
- Cancer
- Cancri
- Cae
- Caelii
- Caelum

Si la préférence *check* est cochée ou lorsque l'utilisateur lance depuis le menu *Name*, *Check found names*, les noms d'objet sont testés un par un dans SIMBAD au moyen du web service `queryObjectById` par la fonction `Annot.checkFoundNames`.

Les noms au format SIMBAD sont calculés en remplaçant l'acronyme du nom détecté par l'acronyme SIMBAD associé aux expressions régulières qui correspondent à la détection. Lorsqu'une détection a plusieurs noms SIMBAD possibles, ils sont tous essayés jusqu'à ce que l'un d'eux existe et devienne son `ObjName.getExistingName`. Il sera modifié si l'utilisateur lui en donne un autre au moment de l'ajouter dans la liste des identificateurs.

Dans l'arborescence, les noms d'objet ont une icône différente selon qu'ils ont un () ou plusieurs () noms SIMBAD possibles d'après le dictionnaire. Par exemple, “NGC 1647” n'a qu'une possibilité alors que “M 31” en a six.

Leur état passe de Non vérifié à Déjà entré, Existant ou Non existant et leur couleur change.

Figure 9. Légende des couleurs des noms d'objet

Etat	Existence	en HTML	en PDF	dans DJIN
Déjà entré	oui	NGC 1404	NGC 1404	NGC 1404
Existant	oui	AC02	AC02	AC02
Rejeté	oui	CH389	CH389	CH389
Non existant	non	CH388	CCD 9	CCD 9
Mauvais format	non	Chiron 389	Chiron 389	Chiron 389
Non vérifié	?	ACCG 02	ACCG 02	ACCG 02

La colonne “Dans DJIN” correspond à la couleur dans l'arborescence, la liste déroulante d'ajout dans les identificateurs ou la liste des identificateurs.

> **Note**
> La couleur de l'état Rejeté est aussi utilisée pour les noms qui proviennent d'une recherche avec préfixe et dont l'existence n'a pas été vérifiée.

Si la préférence *`Automatic detection of wrong object names`* est cochée, le programme tente de rejeter automatiquement les fausses détections.

2.1.9. Rejet automatique des fausses détections

DJIN utilise un moteur d'apprentissage pour tenter de supprimer automatiquement (= mettre dans la liste des rejetés) les noms d'objet qu'il considère comme du bruit.

Lorsque la préférence *`Automatic detection of wrong object names`* est cochée, la fonction `Annot.verifyFalsePositives` tente de deviner pour chaque occurrence si elle doit être considérée comme du bruit et rejetée.

DJIN intègre donc Weka[2] qui est un module d'exploration de données (data mining) contenant des algorithmes d'apprentissage. Ce module est également utilisé par le projet VO-Tech AstroWeka[3].

A partir de données d'entrainement sur des noms d'objet corrects et incorrects un algorithme tente de déterminer si les nouveaux noms sont du bruit ou non.

Les informations disponibles pour chaque nom d'objet sont les suivantes :

- Type de position (titre, résumé, sous-titre, texte, table, ...)
- Nom d'occurences
- Séparateur avant
- Séparateur après
- Longueur du nom
- Longueur de l'acronyme
- Première lettre (ou chiffre) de l'acronyme
- Longueur du nom moins celle de l'acronyme
- Existence de l'objet dans SIMBAD
- La ligne commence-t-elle avec un chiffre
- La ligne finit-elle avec un nombre entre parenthèses (comme les formules mathématiques)

[2] http://www.cs.waikato.ac.nz/~ml/weka/index.html
[3] http://sourceforge.net/projects/astroweka/

- Nombre de mots parmi ceux habituellement présents dans les adresses (la liste est dans `address_word` . Ex: Observatoire, Laboratoire, Institut, Université, cedex, ...)
- Est-ce un nom d'auteur de la liste des références suivi d'une année (référence bibliographique)?
- Nombre de caractères non reconnus ou en indice dans la ligne
- Nombre de mots de la ligne faisant penser qu'il ne s'agit pas d'un nom d'objet (la liste est dans `suspect_word` . Ex: spectral, type, catalog, telescope, ...)
- Nombre de mots de la ligne faisant penser qu'il s'agit bien d'un nom d'objet (la liste est dans `nice_word` . Ex: globular, cluster, galaxy, ...), en corrélation avec le type SIMBAD de l'objet

L'évaluation des noms d'objet est effectuée en dernier après leur vérification dans SIMBAD car le résultat de cette vérification fait partie des données prises en compte.

Les mauvais noms sont retirés de l'arborescence principale et placés dans la liste des noms rejetés avec ceux que l'utilisateur supprime. Ceux qui sont abusivement rejetés peuvent être remis dans l'arborescence principale en cliquant sur *`Accept`* ou par glisser-déplacer.

Plusieurs algorithmes d'apprentissage sont disponibles dans Weka. Celui finalement retenu est : `weka.classifiers.bayes.NaiveBayes`. Il est utilisé avec un méta-moteur et une matrice de coût pour indiquer qu'il vaut (10 fois) mieux laisser passer des faux positifs que rejeter des bons. Cette initialisation se fait dans le constructeur de `LearningMachine` .

Le fichier `names.arff` contient les données d'apprentissage. Il peut être complété de la manière suivante :

- Lancer DJIN avec l'option -training
- Ouvrir un bibcode
- Supprimer les fausses détections
- Lorsque l'arborescence ne contient que des bons noms d'objet lancer la commande *Identifier, Add all names*.

En plus d'ajouter les noms d'objet dans la liste des identificateurs, DJIN va ajouter une ligne par nom d'objet dans le fichier `names.arff` .

> **Tip**
> Lorsque l'on a plusieurs installation de DJIN sur une machine, on peut avoir envie d'ajouter les données d'apprentissage pour les noms d'objet d'un bibcode provenant d'une autre installation. Pour cela, il suffit d'ouvrir le bibcode puis de lancer la commande *Name, Delete names according to an ARFF file* (qui n'est visible qu'avec l'option -training) et de sélectionner le fichier contenant les données d'apprentissage que l'on souhaite importer. Les noms d'objet signalés comme devant être supprimés sont à nouveau supprimés et il suffit de lancer la commande *Identifier, Add all names* pour ajouter les données désirées dans l'installation courante.

2.1.10. Création de document

Un nouveau document est créé lorsque l'utilisateur demande la visualisation du document annoté (*File, View annotated document* ou ) ou demande son enregistrement dans un fichier (*File, Save annotated document*).

Pour les documents PDF, la fonction `Annot.saveDocument` sauvegarde le document PDF courant dans lequel les annotations ont déjà été créées lors de la détection et leur couleur modifiée lors de la vérification par `Annot.setAnnotationColor`. Des marque-pages sont également ajoutés pour pouvoir retrouver les noms facilement. Au passage cette fonction fait sauter les protections que les

auteurs peuvent mettre dans les documents PDF comme l'interdiction d'imprimer ou de copier le contenu.

Les annotations PDF ont un défaut, elles ne s'impriment pas à cause d'un bug dans Acrobat Reader. Pour pouvoir les imprimer quand même, la commande *File, Add printable annotations* dessine des traits de couleur dans le document PDF qui une fois ajoutés ne peuvent plus être retirés.

Pour les documents HTML, la fonction `Annot.saveHtmlDocument` parcourt la liste des noms d'objet détectés et génère une liste de liens HTML vers les noms d'objet dans le fichier `lstAnnot.html` .

La fonction `Annot.underlineInHtml` repère ensuite dans le fichier HTML la position des noms d'objet détectés et insère des balises supplémentaires pour leur mise en valeur. Pour cela la manière dont les caractères spéciaux sont écrits en HTML doit être connue pour permettre de corriger les positions par rapport à celles dans le texte extrait depuis le document HTML. Le fichier `unicode.html` fournit ces informations.

La liste des liens et le document HTML annoté `annot.html` sont référencés dans le document maître `mainAnnot.html` utilisant les frames HTML.

Les noms d'objet mis en valeur en PDF ou HTML sont des liens vers leur description dans SIMBAD dont l'URL est construite à partir de la valeur du paramètre url.

2.1.11. Exportation du texte extrait

Le texte extrait du document courant peut être enregistré dans un fichier au format texte ou HTML.

Par défaut, le répertoire de destination est celui de la préférence utilisateur Destination.

Au format texte, les caractères standards et ceux écrits en petit (notes, légendes, tables et parfois abstract) sont séparés par une ligne de "_____". Au format HTML, les types de position sont écrits comme dans la fenêtre principale de DJIN, et sont à nouveau reconnus lors de la réouverture du document.

L'exportation HTML peut aussi être déclenchée lors du traitement d'une liste de documents si l'option -export est précisée dans la ligne de commande.

2.1.12. Retour au document

La commande *File, Back to document* ou le bouton  permettent de revenir au texte extrait dans la fenêtre principale de DJIN après avoir lancé le chargement du dernier dictionnaire ou affiché le résultat d'une requête à SIMBAD (si la préférence descInBrowser n'est pas cochée).

2.1.13. Sauvegarde de l'état courant

La commande *File, Save current state* permet de sauvegarder l'état complet du programme en local. Il sera récupéré au prochain lancement du programme.

Les éléments suivants sont sauvegardés dans le sous-répertoire `current` :

- Document PDF ou HTML annoté
- Texte extrait
- Liste des identificateurs
- Position des fenêtres
- Contenu des zones de saisie

La fonction `CurrentState.saveState` sauvegarde ces éléments en utilisant la sérialisation des objets de Java. C'est pourquoi on ne peut plus restaurer l'état courant après avoir fait une mise à jour de DJIN, car les définitions des objets implémentant `java.io.Serializable` ont pu changer.

La restauration de l'état courant se fait simplement au démarrage suivant du programme.

Il faut effacer cet état courant avec *File, Clear current state* lorsqu'on en n'a plus besoin.

2.2. Noms d'objet détectés dans les documents

2.2.1. Vérification des noms d'objet dans SIMBAD

Lorsque la case *check* est cochée dans les préférences utilisateurs, les noms d'objet sont automatiquement vérifiés dans SIMBAD après la phase de détection et après chaque nouvelle recherche.

L'utilisateur doit lancer cette vérification manuellement avec *Name, Check found names* si la case n'est pas cochée.

2.2.2. Envoi de requêtes vers SIMBAD

La commande *Name, Request SIMBAD* permet d'obtenir la description d'un objet dont le nom est sélectionné dans l'arborescence. C'est le nom donné par `ObjName.getExistingName` qui est utilisé. On peut aussi l'obtenir en cliquant sur les liens du texte extrait.

La commande *Name, Request bibcode in SIMBAD* permet d'obtenir la description d'une référence bibliographique avec sa liste d'objets. Le bibcode est demandé à l'utilisateur. Si du texte est sélectionné dans le texte extrait, ou est présent dans le presse-papier, il est proposé par défaut. Une tentative pour le transformer en bibcode est effectuée par la fonction `BibcodeListener.fromReference`. Cela permet à l'utilisateur de sélectionner une ligne de la liste des références à la fin de l'article afin d'obtenir sa description.

Les descriptions s'affichent dans un navigateur si la case descInBrowser est cochée ou à la place du texte extrait sinon.

2.2.3. Recherche des noms d'objet déjà entrés

Cette fonctionnalité permet lorsqu'un article a déjà été traité d'en obtenir une version annotée où les noms d'objet sont mis en valeur en tirant profit des information entrées dans SIMBAD (raw id, occurrences, positions) par les documentalistes.

La fonction `SearchEntered.loadSearchHighlight` lance une requête SQL pour charger les informations disponibles sur la référence, puis désactive l'arbre de recherche pour pouvoir effectuer une recherche par expression régulière simple pour chaque nom d'objet à retrouver. Ils seront tous soulignés en vert qui est la couleur des noms déjà présents sur la référence.

Cette mise en valeur peut être effectuée pour des listes de documents en utilisant l'option -enteredNames.

Exemple : `./parfileReader.sh -edit A\&A -list /tmp/aa.txt -enteredNames`

2.2.4. Suppression d'un nom d'objet, liste des noms d'objet rejetés

La suppression d'un nom d'objet avec la touche **Suppr** l'envoie dans la liste des noms d'objet rejetés. Il peut aussi y être glissé avec la souris.

La classe `ObjNameTransferHandler` contient les fonctions de glisser-déplacer entre les listes de noms d'objet. A chaque transfert, les noms d'objets sélectionnés dans la liste de départ sont déplacés dans la liste de destination.

La commande *Name, Display rejected names* ou le bouton  affiche et masque la liste des noms d'objet rejetés.

> **Note**
> Seuls les noms d'objet de la liste principale peuvent être ajoutés dans la liste des identificateurs.

2.2.5. Les noms d'objet indéterminés

La liste des noms d'objet indéterminés peut être utilisée pour mettre de côté certains noms à traiter plus tard. Elle est contenue dans l'objet `SearchName.lstUndetermined`.

Elle peut être affichée et masquée avec la commande *Name, Display undetermined names* (ou le bouton ).

Les noms d'objet peuvent être placés dans cette liste par glisser-déplacer ou la commande *Name, Mark selected names as undetermined*.

2.2.6. Les noms d'objet traités

La liste des noms d'objet traités peut être utilisée pour mettre de côté les noms déjà traités. Elle est contenue dans l'objet `SearchName.lstDone`.

Elle peut être affichée et masquée avec la commande *Name, Display done names* (ou le bouton ).

Les noms d'objet peuvent être placés dans cette liste par glisser-déplacer ou la commande *Name, Mark selected names as done*.

2.2.7. Edition de la liste des noms d'objet

La commande *Name, View name list* permet d'afficher dans un document PDF les différentes listes de noms d'objet détectés dans un document avec leur nom SIMBAD et leurs nombres d'occurrences pour les différents types position.

La fonction `PDFList.savePdfDocument` génère le fichier `names.pdf` en utilisant les fonctions de création de document PDF de la bibliothèque PDFBox[4].

2.3. Identificateurs, mise à jour de SIMBAD

Les identificateurs sont les noms des objets qui seront ajoutés à la référence bibliographique dans SIMBAD.

2.3.1. Ajout de noms d'objet

Les liste des identificateurs est remplie à partir de noms d'objet détectés par DJIN (ou l'utilisateur).

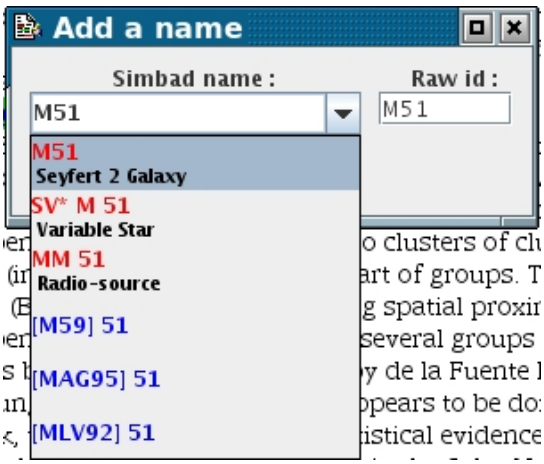
Les noms d'objet de l'arborescence peuvent être ajoutés un par un, ou tous ensemble ou seulement ceux sélectionnés grâce aux commandes du menu *Identifier*.

La commande *Identifier, Add from clipboard* ajoute le texte du presse-papier séparé par les retours chariot. Dans ce cas, il n'y pas d'informations sur le nombre d'occurrences et les types de position.

[4] http://www.pdfbox.org

Pour un ajout simple ou lorsqu'il y a plusieurs noms SIMBAD possible la boîte de dialogue d'ajout d'un nom est affichée (sauf si la préférence *`Don't ask for ambiguities when adding several names`* est cochée, ce qui est déconseillé).

Figure 10. Boîte de dialogue d'ajout d'un nom d'objet



A l'ouverture de cette boîte, la fonction `AddNameDlg.init` lance en parallèle dans un thread l'interrogation de SIMBAD avec les noms de la liste déroulante pour pouvoir les afficher dans la couleur correspondant à leur état et écrire leur type principal en-dessous. Pour de longues listes, des noms peuvent apparaître en noir le temps que la réponse arrive et qu'ils soient réaffichés.

Pour remplir la liste déroulante, une liste d'objets `SimbadName` est utilisée. Ces objets stockent le nom, l'acronyme, l'acronyme de renvoi et le score de noms d'objets astronomiques SIMBAD. La liste est triée pour que les noms avec renvoi soient placés après ceux sans renvoi (car ils sont moins suceptibles d'être utilisés). Les noms sont ensuite classés en fonction de leur score.

Les scores sont en fait la fréquence avec laquelle ils ont été utilisés par les auteurs d'après les raw id stockés dans SIMBAD sur une période donnée (initialement : l'année 2008). Ces fréquences sont enregistrées dans le fichier `acro_freq` pour chaque acronyme de raw id.

Lorsqu'un objet possède plusieurs composantes (ex. : ROXs 42), il est retiré de la liste et les composantes sont proposées à la place.

Par exemple, le nom d'objet détecté “ADS 7” donne comme noms SIMBAD possibles “** ADS 7” et “ADS 7”. Le premier n'existe pas et le second est un nom à composante qui retourne les possibilités : “** ARG 47”, “ADS 7AB” et “HD 240479”. La liste finale sera donc :

- ** ADS 7
- ** ARG 47
- ADS 7AB
- HD 240479

Pour éviter de poser trop de fois des questions similaires, les acronymes des noms proposés et effectivement ajoutés lors de la question précédente sont conservés en mémoire. Si le même acronyme revient, la case *`Remember this choice`* apparaît et permet d'éviter de poser la question une 3ème fois. La fonction `ActionAddName.addNameDialog` se souvient également lorsque l'utilisateur valide un nom sans le modifier.

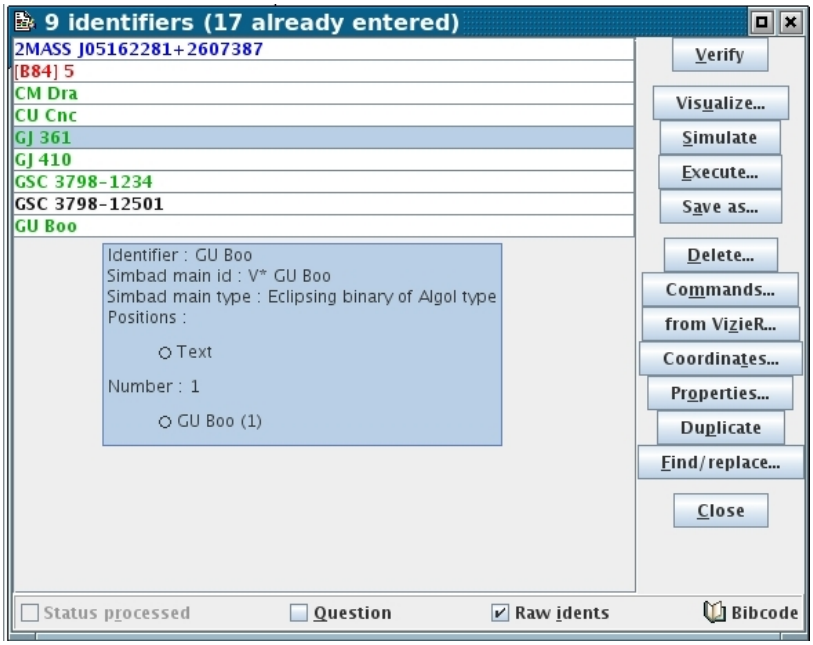
Après un ajout multiple, la liste des identificateurs est affichée.

2.3.2. Effacement de la liste des identificateurs

La commande *Identifier, Clear identifiers* efface complètement la liste des identificateurs contenue dans l'objet liste `Menu.lstIdent`.

2.3.3. Liste des identificateurs

Figure 11. Liste des identificateurs



2.3.3.1. Vérification dans SIMBAD

Le bouton *`Verify`* lance `Identifiers.verifyInSimbad` qui vérifie les identificateurs dans SIMBAD. Leur couleur est modifiée en conséquence et les informations sur l'objet astronomique (type, identificateur principal) sont stockées dans l'objet `Ident` .

La fonction `Identifiers.testUnicity` compare les identificateurs principaux et regroupe les différents noms d'un même objet en conservant les raw id et en additionnant les nombres d'occurrences. Il est donc utile de lancer la vérification même lorsque tous les objets sont coloriés comme existant (rouge ou vert) pour regrouper les doublons.

A chaque modification de la liste, la fonction `Identifiers.arrange` regroupe également les lignes contenant le même nom d'identificateur.

Lorsque la vérification d'un identificateur tombe sur un objet à plusieurs composantes, la boîte de dialogue d'ajout d'un identificateur est affichée pour permettre à l'utilisateur de choisir la bonne composante.

Le bibcode est également vérifié, ce qui fait passer son icône de  à  et permet d'afficher dans une bulle des informations telles que le titre, les auteurs, etc...

2.3.3.2. Propriétés des identificateurs

Les noms des identificateurs peuvent être modifiés directement dans la liste. Le passage en mode édition se fait en appuyant sur **F2** ou par double clic.

La liste gère la multi-sélection. On peut regrouper les identificateurs en faisant glisser la sélection vers une ligne donnée. Toute la sélection est alors regroupée sous le nom de l'identificateur destination. La classe `IdenListTransferHandler` gère le glisser-déplacer dans cette liste.

Les lignes sélectionnées peuvent être supprimées avec *`Delete`* ou dupliquées avec *`Duplicate`* (les raw id et nombres d'occurrences sont également dupliqués).

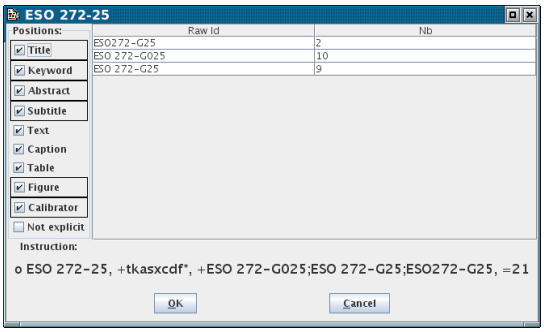
Le bouton *commands* permet d'afficher et modifier des commandes de mise à jour supplémentaires sur chaque identificateur (`Ident.getCommands`). Une requête par position peut être envoyée à SIMBAD avec le bouton *Position* à condition que la position soit connue (l'objet existe) ou qu'une commande de modification des coordonnées fasse partie des commandes de mise à jour.

Le bouton *from Vizier* (disponible aussi dans la fenêtre des commandes) permet d'interroger l'utilitaire `gsc4sim` pour obtenir des commandes qui sont ajoutées à chaque identificateur sélectionné. Seuls les identificateurs dont le format correspond à une des expressions régulières du fichier `gsc4sim_format` sont envoyés par la fonction `IdentDlg.gsc4sim`.

Le bouton *Coordinate* lance la fonction `IdentDlg.coordinates` qui recherche des coordonnées dans les noms des identificateurs sélectionnés. Une commande d'ajout de coordonnées est alors ajoutée aux commandes de mise à jour de l'identificateur.

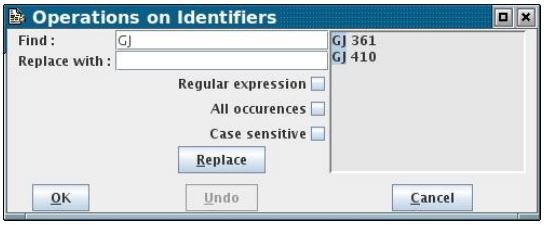
Le bouton *Properties* affiche une boîte de dialogue (classe `PropertiesDlg`) permettant d'ajouter des types de position et d'éditer la liste des raw id avec leurs occurrences pour les identificateurs sélectionnés. Si la sélection contient plusieurs lignes, les raw id ne peuvent pas être modifiés mais un nombre commun peut être ajouté aux nombres d'occurrences (sur les raw id qui en ont déjà le plus).

Figure 12. Propriétés d'un identificateur de la liste



Le bouton *Find/replace* affiche une boîte de dialogue (classe `TransformDlg`) permettant d'effectuer des opérations de remplacement sur des listes d'identificateurs sélectionnés :

Figure 13. Transformation de listes d'identificateurs



La liste des identificateurs sélectionnés est reproduite dans la partie droite (la fenêtre est redimensionnable). Les caractères correspondant au format saisi dans la zone *Find:* sont sélectionnés pour les mettre en valeur.

Le remplacement avec le texte de la zone *Replace with:* est effectué à chaque clic sur le bouton *Replace* (*Undo* permet de revenir en arrière).

La case *Regular expression* permet d'utiliser les expressions régulières. Un rappel de la syntaxe est alors disponible dans une bulle au-dessus de l'icône . La case *All occurrences* permet de remplacer toutes les occurrences de chaque ligne et pas seulement la première.

Les changement sont reportés dans la liste des identificateurs sélectionnés après avoir appuyé sur *OK*.

2.3.3.3. Propriétés du bibcode

Les propriétés suivantes concernent la référence bibliographique sans être sur un objet particulier.

- *Raw idents* (cochée par défaut) indique qu'il faut inclure les informations collectées par DJIN aux commandes de mise à jour (raw id, positions, nb d'occurrences)
- *Status processed* supprime “=0=” du commentaire de travail (grisé si “=0=” est absent)
- *Question* ajoute “=q=” au commentaire de travail
- *Bibcode* permet de saisir des commandes de mise à jour pour le bibcode. Elles sont stockées dans `Identifiers.getStrBibCmd`.

2.3.3.4. Opérations sur les scripts

Visualize affiche le script (généré par la fonction `Identifiers.writeCommands`) dans une fenêtre.

Le bouton *Simulate* génère une version de ce script où les commandes “v” de validation sont remplacées par des “q!” et l'exécute en utilisant le programme de mise à jour et le login spécifiés dans les préférences utilisateur.

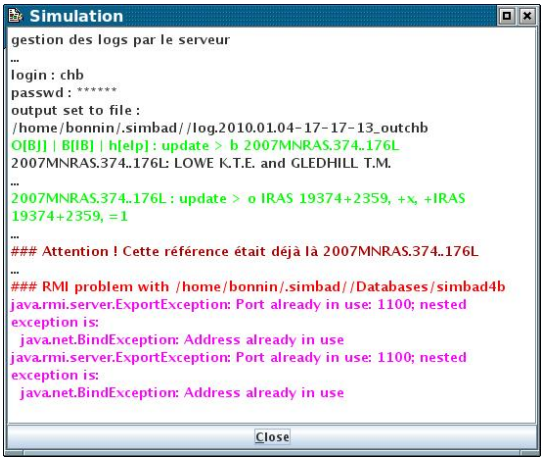
> **Warning**
> Si des commandes “v” ou “w” sont présentes dans les commandes de mise à jour des identificateurs ou du bibcode, la simulation effectuera quand même des modifications dans SIMBAD.

Execute utilise les mêmes préférences pour exécuter le script tel qu'il est affiché par *Visualize*.

Le résultat des exécutions (les sorties erreur et standard) sont affichées en noir sauf :

- la répétition des lignes du script à exécuter sont en vert
- les lignes commençant par “###” et indiquant que la référence était déjà là sont en rouge sombre
- les autres lignes commençant par “###” sont en rouge vif
- les lignes contenant un message d'exception Java sont en magenta

Figure 14. Exemple d'exécution d'un script de mise à jour de SIMBAD



Le bouton *Save As* enregistre le script de mise à jour sous un nom demandé à l'utilisateur, le répertoire par défaut étant celui de la préférence identifier

Ce bouton devient *Open script* si la liste des identificateurs est vide et permet d'ouvrir un script de mise à jour précédemment enregistré. DJIN enregistre automatiquement le script de mise à jour avant chacune des actions suivantes afin de sauvegarder le travail du documentaliste : (Visualisation ou exécution, Vérification des identificateurs, interrogation de `gsc4sim` depuis la fenêtre des commandes, sauvegarde de l'état courant, sortie du programme). Le répertoire par défaut est le sous-répertoire `script` .

2.3.4. Création d'un nouveau bibcode

L'objectif est de générer les commandes de création d'une nouvelle référence bibliographique dans SIMBAD. Le script résultat contient le bibcode complet, le numéro de dernière page, le titre, les auteurs et ajoute le commentaire de travail “=0=”.

L'utilisateur copie dans le presse-papier un texte contenant toutes ces informations. Les premières lignes du document PDF contenant l'article conviennent en général mais l'article en HTML peut également être utilisé. Puis il lance la commande *Identifier, Create bibcode from clipboard*.

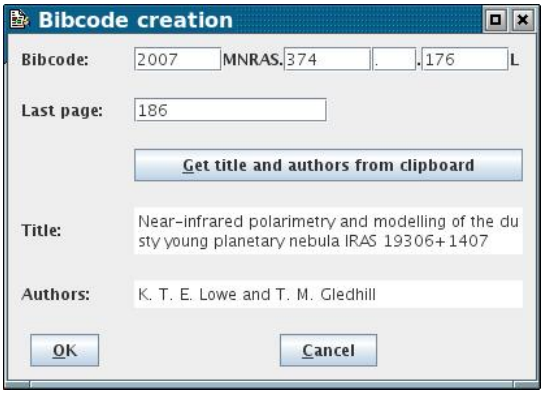
Le format de ce texte doit au préalable être enregistré dans le paramètre bibcodestring sous la forme d'une expression régulière. Il doit indiquer l'emplacement des différentes informations par leur lettre correspondante précédée de “%”:

- %y = year
- %v = volume
- %f = first page
- %l = last page
- %t = title
- %a = authors

Au cas où le texte du presse-papier contiendrait des caractères **%**, ils peuvent être codés dans le format par “%%”.

Si un champ manque, l'utilisateur devra le saisir dans la boîte de dialogue de création de bibcode.

Figure 15. Création d'un nouveau bibcode



Une fois le texte dans le presse-papier, la commande *Identifier, Create bibcode from clipboard* ouvre la boîte de dialogue et la fonction `NewBibcodeDlg.init` applique l'expression régulière générée à partir du format dans laquelle les champs ont été remplacés par des groupes. Si le format correspond, le contenu des groupes est récupéré et les différentes informations sont affichées dans les zones de saisie correspondantes.

Voici un exemple de format :

Mon. Not. R. Astron. Soc. %v, %f.%l \(%y\).\n%t%a*

Les caractères ayant une signification en expression régulière doivent être précédés de “\”. “\n” représente un retour chariot. Il est inutile de mettre des retours chariots après les champs titre et auteur car ces informations peuvent être sur plusieurs lignes. Le programme tente de les séparer en repérant les initiales des noms d'auteur.

Les caractères indésirables sont supprimés et les caractères accentués sont remplacés par leur caractère de base en utilisant le contenu du fichier `accented_ascii` .

Le bouton *Get title and authors from clipboard* récupère à nouveau le contenu du presse-papier pour n'y rechercher que le titre suivi des noms d'auteurs, ce qui permet d'entrer les informations en deux fois si nécessaire.

La dernière lettre du bibcode est déterminée automatiquement d'après le premier auteur. Si c'est l'initiale du prénom qui est choisie à tort, il faut modifier ce prénom en ne gardant que l'initiale suivie d'un point pour que le programme détermine correctement la fin du bibcode.

Les commandes de mise à jour sont générées après avoir appuyé sur *OK* et accessibles depuis la liste des identificateurs. A ce stade, il vaut mieux exécuter sans ajouter d'identificateurs car le contenu de l'article ne peut être récupéré par DJIN à partir du bibcode que quand celui-ci existe dans SIMBAD.

Les nom d'auteurs sont convertis (par la fonction `NewBibcodeDlg.formatAuthors`) pour être écrits sous la forme : NOM P.

Les possibilités de départ étant : Prenom Nom, Nom P., P. Nom, J.-Ph. Nom, P. Q. Nom

2.4. Recherches supplémentaires

Les recherches supplémentaires (sauf celle dans le texte extrait) ajoutent des détections de noms d'objet à celles faites par la recherche principale.

2.4.1. Recherche dans le texte extrait

Pour rechercher et sélectionner simplement un mot dans le texte extrait sans le considérer comme un nom d'objet.

La commande *Search, Search in the extracted text* ouvre la boîte de dialogue `SearchDlg` qui déplace le curseur dans le texte extrait vers la prochaine occurrence du texte cherché par l'utilisateur.

2.4.2. Recherche avec préfixe

L'objectif est de relancer la recherche en précisant un préfixe qui sera ajouté à tous les mots du texte pour trouver des noms dont l'auteur a systématiquement omis le début.

Par exemple, si l'auteur précise seulement 0235+164 au lieu de QSO B0235+164, le nom peut être trouvé en effectuant une recherche avec le préfixe QSO B.

La fonction `Annot.prefixSearch` relance la recherche après avoir extrait de l'arbre de recherche [11] principal le sous-arbre correspondant au préfixe demandé.

Warning

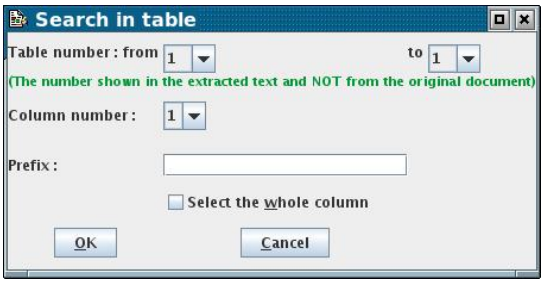
Pour certains préfixes, cette recherche peut générer beaucoup de bruit. Par exemple le préfixe “NGC ” considèrera tous les chiffres comme étant des noms d'objet du catalogue NGC.

2.4.3. Rechercher dans une table

La commande *Search, Search in a table* permet d'effectuer une recherche avec ou sans préfixe limitée à une colonne d'une ou plusieurs tables du texte extrait. Ces tables sont détectées et numérotées par

l'extraction de texte. Les numéros dans DJIN peuvent ne pas correspondre avec ceux de l'auteur comme dans le cas où une table est à cheval sur plusieurs page, elle n'est comptée qu'une fois dans l'article et plusieurs fois dans DJIN.

Figure 16. Recherche de noms d'objet dans les tables



L'utilisateur précise les tables et la colonne désirées. Le préfixe est particulièrement utile lorsque l'auteur ne l'écrit que dans l'entête de colonne du tableau.

La case *Select the whole column* permet de prendre comme nom d'objet tout le contenu de la colonne sans tenir compte des formats.

La fonction Annot.tableSearch lance une recherche en se limitant aux lignes du texte extrait de type table et en comptant les tables pour ne garder que les numéros demandés.

2.4.4. Recherche élargie

Cette recherche a été conçue pour ajouter aux noms déjà détectés tout ce qui pourrait ressembler à un nom d'objet en se basant sur la liste d'expressions régulières contenues dans le fichier regexpLarge

La fonction SearchName.searchFileInit initialise donc la recherche pour utiliser la liste d'expressions régulières à la place de l'arbre de recherche.

2.4.5. Ajouter une expression régulière

Il est possible d'augmenter le nombre de noms trouvés par la recherche principale.

La commande *Search, Add a regular expression* permet d'ajouter une expression régulière à l'arbre de recherche (fonction SearchName.astroAddRegExp). Elle sera donc utilisée pour toutes les futures recherches mais sera oubliée à la sortie du programme. Par contre, DJIN ajoute automatiquement au démarrage les expressions régulières contenues dans le fichier listeRegExp [47] .

Les expressions régulières ajoutées, y compris celles du fichier, peuvent être supprimées avec *Search, Remove a regular expression*.

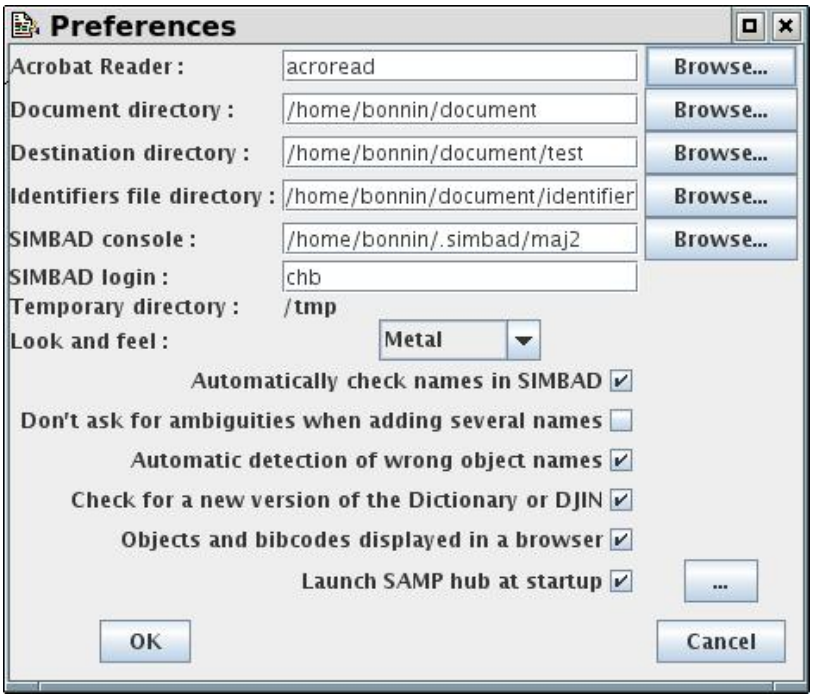
Un historique des expressions régulières ajoutées est conservé en local dans la préférence prefRegex. Les expressions du fichier listeRegExpHistory [47] sont automatiquement ajoutées à cet historique.

Une recherche standard est lancée après chaque ajout ou suppression dans l'arbre de recherche.

2.5. Configuration du programme

2.5.1. Modification des préférences de DJIN

Figure 17. Préférences



Les préférences sont destinées à être modifiées par l'utilisateur et sont spécifiques à son poste de travail.

Elles sont stockées en local sur le poste au moyen de l'API Java `java.util.prefs.Preferences`.

Le noeud utilisé est celui de la fenêtre principale : `cds.ObjectName.Ihm.Menu`.

Table 1. Préférences enregistrées sur le poste utilisateur

Paramètre	Type	Explication	Exemple
acrobat	Chaîne	Chemin du programme de visualisation des fichier PDF	/usr/bin/acroread
ambigu	booléen	L'utilisateur n'est pas consulté lorsqu'un nom est ambigü d'après le Dictionnaire de Nomenclature	0
check	booléen	Les noms d'objet doivent-ils être testés automatiquement dans SIMBAD ?	1
descInBrowser	booléen	Les description des noms d'objet et d'article par SIMBAD sont affichées dans le navigateur plutôt que dans DJIN	true
destination	Chaîne	Chemin du répertoire où enregistrer les documents annotés	/home/.../document/ test
dictionary	booléen	Faut-il rechercher une nouvelle version du Dictionnaire de Nomenclature	1
document	Chaîne	Chemin du répertoire par défaut pour les documents PDF	/home/.../document
identifier	Chaîne	Chemin du répertoire par défaut d'enregistrement des scripts de mise à jour	/home/.../document/ identifier
learning	Booléen	Utilisation du moteur d'apprentissage pour détecter automatiquement les noms d'objet à rejeter	1
login	Chaîne	Login utilisé pour l'exécution et la simulation des scripts de mise à jour	chb
look	Chaîne	Style d'affichage du programme	javax.swing.plaf.metal. MetalLookAndFeel
samphub	Booléen	Lancement du hub SAMP au démarrage de l'application pour recevoir les messages en provenance de Firefox	false
simbad	Chaîne	Chemin du programme de mise à jour de SIMBAD	/home/.../.simbad/maj2

Table 2. Autres réglages enregistrés sur le poste utilisateur

Paramètre	Type	Explication	Exemple
boldFont	booléen	Graisse de la police choisie pour la visualisation des scripts de mise à jour	true
dicoDate	Date	Date de la dernière version du Dictionnaire de Nomenclature	23-mai-2008
fontName	Chaîne	Nom de la police de visualisation des scripts de mise à jour	Dialog
fontSize	Entier	Taille de la police choisie pour la visualisation des scripts de mise à jour	12
italicFont	Booléen	Police choisie pour la visualisation des scripts de mise à jour en italique	false
regex	Chaîne	Expressions régulières proposées dans l'historique lors de l'ajout d'une expression régulière (séparés par des ' ~ ') en plus de celles ajoutées par le fichier `listeRegExpHistory` [47]	SDSS_J[-+0-9.] {10,20} ~ \d{4,10} (\.\d{1,2})? ?[+\-] ? \d{2,10}(\.\d{1,2})? ~ \d{5}[+\-]\d{4}
windowHeight	Entier	Hauteur de la fenêtre principale lors de la dernière exécution du programme	629
windowWidth	Entier	Largeur de la fenêtre principale lors de la dernière exécution du programme	840
windowX	Entier	Abscisse de la fenêtre principale lors de la dernière exécution du programme	328
windowY	Entier	Ordonnée de la fenêtre principale lors de la dernière exécution du programme	200

- La zone *`Acrobat reader`* permet d'indiquer le chemin du programme de visualisation des documents PDF
- La zone *`Document directory`* contient le répertoire d'ouverture par défaut des fichiers par DJIN
- La zone *`Destination directory`* contient le répertoire d'enregistrement par défaut des fichiers (enregistrement du document annoté, exportation du texte extrait)

- La zone *Identifiers file directory* contient le répertoire d'enregistrement par défaut des fichiers de commandes de mise à jour de SIMBAD.

- La zone *SIMBAD console* permet d'indiquer le chemin du programme de mise à jour de SIMBAD utilisé pour exécuter [21] ou simuler [21] les scripts de mise à jour de SIMBAD.

- La zone *SIMBAD login* contient le nom d'utilisateur (login) pour lancer cette mise à jour

- Le répertoire temporaire s'affiche de manière non modifiable. Il est utilisé pour les fichiers temporaires (transformations XSL, fichiers de commandes de mise à jour, document annoté PDF ou HTML, fichiers téléchargés, ...).

- La liste déroulante *Look and feel* permet de sélectionner le thème utilisé pour dessiner les fenêtres

- La case à cocher *Automatically check names in SIMBAD* indique si les noms d'objet doivent être automatiquement vérifiés dans SIMBAD après la phase de détection dans le texte extrait

- La case à cocher *Don't ask for ambiguities when adding several names* indique à DJIN de ne pas poser de question à l'utilisateur lors de l'ajout d'un nom d'objet dans la liste des identificateurs. Dans ce cas le premier identificateur existant trouvé est choisi comme identificateur du nom d'objet. (Il est déconseillé de cocher cette case !).

- La case à cocher *Automatic detection of wrong objet names* indique à DJIN de tenter de déterminer parmi les noms d'objet détectés les fausses détections et de les transférer automatiquement dans la liste des noms rejetés.

- La case à cocher *Check for a new version of the Dictionary or DJIN* indique si la disponibilité d'une nouvelle version du Dictionnaire de Nomenclature ou de DJIN doit être vérifiée. La possibilité de mettre à jour est alors proposée à l'utilisateur.

- La case à cocher *Objects and bibcode displayed in a browser* indique si les descriptions d'objet ou de bibcodes sont affichées dans un navigateur ou sinon dans la fenêtre principale de DJIN.

- La case à cocher “Launch SAMP hub at startup” indique si le hub SAMP intégré à DJIN doit être lancé au démarrage du programme. S'il n'est pas lancé, DJIN tente de se connecter à un hub existant pour recevoir les messages SAMP [36].

- Le bouton noté “...” lance un moniteur SAMP dans une autre fenêtre (dont le parent est la fenêtre principale pour pouvoir fermer les préférences) qui affiche la liste des programmes respectant le protocole SAMP.

Tip

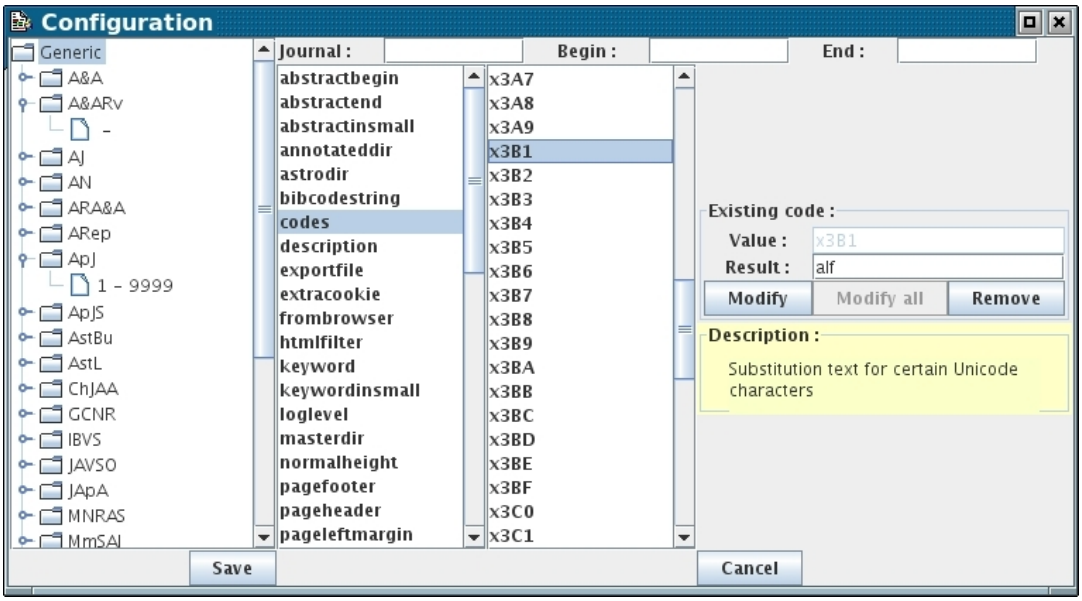
Une fonctionnalité cachée permet de lister dans un navigateur la liste des préférences : Il faut appuyer sur **p** lorsque le bouton *OK* a le focus.

2.5.2. Modification des paramètres d'extraction

Ces paramètres sont surtout utilisés lors de l'extraction de texte depuis un document PDF, mais certains modifient d'autres comportements de DJIN. Ils ne sont pas destinés à être modifiés par l'utilisateur, les préférences étant là pour ça.

Ils sont accessibles par le menu *Configuration*, *Parameters* et sont stockés dans le fichier `config.xml`

Figure 18. Paramètres de configuration



La classe Config lit le fichier et stocke les valeurs des paramètres, tandis que ConfigTree lit le même fichier pour obtenir l'arborescence de tous les journaux et l'enregistre après modifications.

Les descriptions des paramètres sont contenues dans le fichier `parameters.txt` pour pouvoir être affichées dans la fenêtre des paramètres lorsqu'ils sont sélectionnés.

Les paramètres pour un journal et un intervalle de numéros de volume donné sont contenus dans une configuration. Les configurations sont affichées dans l'arborescence de gauche regroupées par journal. La racine de l'arborescence représente la configuration générique à utiliser lorsque le journal est indéterminé

La liste du milieu contient les paramètres. Lorsque l'un d'eux est sélectionné, sa valeur s'affiche dans la zone *`Value`*. Lorsque ces paramètres sont des listes (codes, symbols, positionweights), leurs éléments sont affichés dans la liste de droite, le nom de l'élément dans la zone *`Value`* et le résultat correspondant dans la zone *`Result`*.

Nom	**Description**
abstractbegin	Regular expression that matches the first abstract line
abstractend	Reguler expression that matches the first line after the abstract paragraph
abstractinsmall	true if the abstract paragraph is written in small characters like notes and figure caption
annotateddir	Directory where the annotated documents are written out during the validation of a file list
astrodir	The directory where are the data files (regular expressions list, greek letters, ...)
bibcodestring	Format of the string used to create a new bibcode. It is a regular expression containing the following fields : %y = year, %v = volume, %f = first page, %l = last page, %t = title, %a = authors.
codes	Substitution text for certain Unicode characters
description	Description of the configuration
embeddedsymbols	if true, the characters of incorporated fonts that are not graphically recognized are showed so that the user can enter their meaning (do not enter meaning for characters that never appear in object names)
exportfile	File where all object names from the documents are written out during a file list validation (including the not found names)

Nom	Description
extracookie	Content of the extra cookie required by some sites (ex : MNRAS) to download the PDF documents preceded by a semi-colon
frombrowser	If true, the HTML documents have to be retrieved from Firefox using the SAMPDocument add-on. The "HTML from bibcode" command opens Firefox and the user has to select "Send to DJIN" in the tools menu so that DJIN can process it.
generic	Generic configuration parameters
htmlfilter	Name of XSL filter file that will extract the article content from the HTML document
keyword	Keyword that begins the keyword paragraph
keywordinsmall	true if the key word paragraphi is written in small characters like notes and figure captions
loglevel	Level of java information messages (debug). Possibles values : OFF, INFO, WARNING, ...
masterdir	Directory where original PDF document are copied when validating a list of bibcodes.
normalheight	Minimal height of normal text characters
pagefooter	page footer vertical coordinate
pageheader	page header vertical coordinate
pageleftmargin	left margin horizontal coordinate
pagerightmargin	right margin horizontal coordinate
percentbigsmall	Maximal percentage of big characters allowed in a line for it to be considered as small text
positionweights	Weights for position types (t=title, a=abstract, k=keywords, s=subtitle, c=caption, d=table, f=figure, x=text). They are used when DJIN is looking for keywords in a list of documents to calculate vectors (one line for each bibcode, one column for each group of keywords). The default for undefined values is 1.
ratiosymbol	Maximal ratio of different pixels allowed for two graphic symbols to be considered as equal
reference	Regular expression that matches the begin of the references paragraph
referenceinsmall	true if the references are written in small characters
show_height	Add annotations into the document to display the height and coordinates of characters (debug)
showothersymbol	if true, displays the codes of characters from incorporated fonts that are not graphically recognized nor substituted
specific	Specific configuration parameters
statistic	File where the five words before and five words after each verified object name are written out during validation
subtitleheight	Minimal height of subtitle characters
subtitle	Regular expression that matches a subtitle line
superscript	ratio of character height beyond which the next character will be considered as superscript
symbols	Substitution text for codes of characters from incorporated fonts that are not graphically recognized
tableinsmall	true if the tables are written in small characters like notes and figure captions

Nom	Description
taggeddir	Directory where the extracted texts are written out and the verified object names are tagged during the validation of a file list
titleheight	Minimal height of title characters
url	URL used to create web links in PDF and HTML documents
urlbibcode	URL used to check and complete the bibliographic codes
urlcommand	Command line (e.g. wget) that will be executed to retrieve a PDF document. The %s parameter will be replaced by the URL that points to the document. Arguments must not be surrounded by quotes but just separated by spaces.
urlcoordinate	URL used to perform a search by coordinate in SIMBAD
urlbibcodedesc	URL used to get the description of a bibliographic code (simbo)
urldownload	URL used to download the PDF documents designated by a bibliographic code
urldownload2	Another URL used to download the PDF documents designated by a bibliographic code if an exception occured while using the first URL
urldownloadhtml	URL used to download the HTML documents designated by a bibliographic code
urlmainfile	URL used to download the main and cluster files from the dictionary of nomenclature
wordspacing	ratio of space character width that will be considered as a blank
wordspacingcolumn	Number of spaces that will be considered as a blank between two table columns

2.5.3. Paramétrage d'un nouveau type de document

Voici la procédure à suivre pour le paramétrage d'un nouveau journal dans DJIN.

Les paramètres d'extraction sont variables d'un journal à l'autre, en particulier les tailles des caractères et les dimensions d'une page.

La première chose à faire est d'ajouter un journal dans la fenêtre de configuration (menu *Configuration, Parameters*).

Cliquer sur la 1ère ligne de l'arborescence des configurations. Dans la zone `Journal`, saisir le nom du journal (les zones `début` et `fin` permettent d'entrer un numéro de volume de début et de fin au cas où les paramètres dépendraient aussi du numéro de volume).

Cliquer sur `Add` dans le cadre `Configuration`. Le nouveau nom de journal apparaît dans l'arborescence. Cliquer dessus puis sur la ligne de configuration.

Les paramètres ont alors tous une valeur par défaut. Pour modifier cette valeur, il faut cliquer sur le paramètre, entrer la valeur dans la zone `Value` et cliquer sur `Modify`.

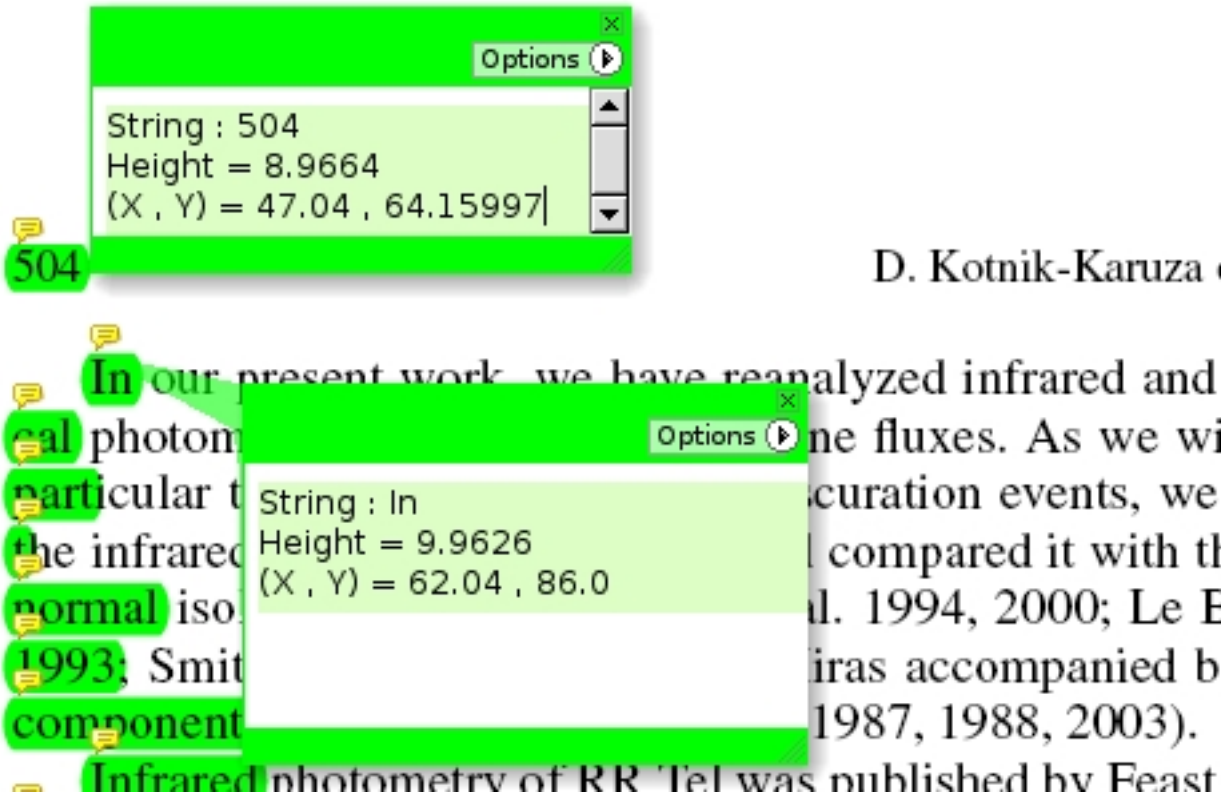
Les paramètres pagefooter et pageheader requièrent une coordonnée en pixel. Les paramètres normalheight, subtitleheight, titleheight requièrent une hauteur de police en pixel. Pour connaître les valeurs à saisir le paramètre show_height commande l'insertion dans un document existant d'annotations qui renseignent sur les coordonnées et hauteurs des groupes de caractères.

Cliquer sur le paramètre show_height et mettre sa valeur à true. Enregistrer la configuration avec le bouton `Save`. Choisir le nouveau journal dans la liste déroulante des journaux, ouvrir un document et le visualiser dans Acrobat Reader.

Dans l'exemple suivant, on voit les coordonnées des caractères de l'entête de page qui ne devra pas être extrait et celles d'un groupe de caractères du texte à extraire. On voit également la hauteur des caractères du texte. Les valeurs des paramètres pageheader, et normalheight seront par exemple respectivement : 75.0 et 9.0

Cette méthode permet de déterminer la valeur des paramètres pageheader, pagefooter, normalheight, subtitleheight (c'est souvent la même) et titleheight.

Figure 19. Extrait de document annoté grâce au paramètre show_height



Le paramètre showothersymbol doit être mis à **true** si des caractères ~ apparaissent à la place du texte extrait, ce qui est le cas lorsque tous les caractères utilisent une police incorporée au document et décrite graphiquement.

Penser à remettre le paramètre show_height à false lorsque toutes les valeurs sont configurées.

2.5.4. Symboles graphiques non reconnus

Les documents PDF peuvent contenir des symboles qui sont décrits seulement graphiquement et dont le code Unicode[5] ou Ascii n'est pas précisé.

Lors de l'extraction de texte depuis un document PDF, les caractères non alphanumériques sont gérés de la façon suivante :

Lorsque le caractère est issu d'une police incorporée dans le fichier PDF dont les caractères sont décrits graphiquement, la fonction `Extract.recognizeSymbol` crée une image du caractère et compte les pixels qu'elle a en commun avec les images du sous-répertoire correspondant au journal dans le répertoire `symbols` . Cette description graphique étant en amont des transformations que peuvent subir les caractères (couleurs, taille, graisse, inclinaison), les pixels peuvent être comparés directement sans recourir à un algorithme de reconnaissance optique (OCR).

Si aucune des images ne correspond, la description de ce symbole dans le fichier PDF peut-être utilisée (si le paramètre showothersymbol est à *`true`*) ou sinon il est remplacé par ~ qui représente

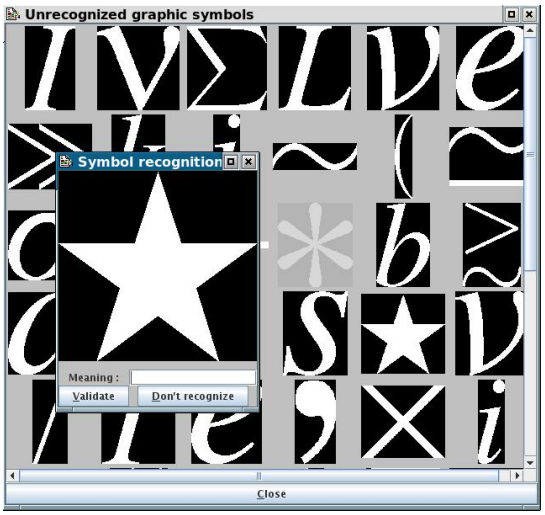
[5] http://www.fileformat.info/info/unicode/

un caractères inconnu. Les descriptions de caractères sont remplacées par leur valeur correspondante si elles se trouvent dans la table des symboles contenue dans les paramètres d'extraction.

Si le caractère est normal mais qu'il n'est pas dans la liste des caractères ASCII (code supérieur à 255), une valeur de substitution est recherchée dans le paramètre d'extraction codes. Si aucune n'est trouvée le caractère est affiché sous la forme `\uXXXX` (il n'est pas censé faire partie d'un nom d'objet).

La liste des caractères non reconnus d'un document PDF peut être consulté par la commande *Config, Display unrecognized graphic symbols* qui ouvre la fenêtre `UnknownSymbolsDlg` .

Figure 20. Symboles graphiques non reconnus d'un document PDF



L'utilisateur peut alors cliquer sur un des symboles et lui attribuer une signification (dans la fenêtre `DisplaySymbolDlg`). Son image est enregistrée dans le sous-répertoire adéquat du répertoire symbols et le symbole est affiché en grisé. L'image sera utilisé dès la prochaine extraction. La correspondance entre les noms des fichiers image et la signification donnée par l'utilisateur est enregistrée dans un fichier nommé `meaning.txt`.

Warning

La mise à jour de DJIN remplace les images stockées sur les postes utilisateur par celles de l'installation qui a servi à créer le paquet de mise à jour.

2.5.5. Création de la liste des expressions régulières

La commande *Configuration, Formats* permet de regénérer le fichier `liste` [46] qui contient les expressions régulières de l'arbre de recherche principal ainsi que leurs acronymes principal et SIMBAD. Pour cela, la fonction `RegularFactory.createList` télécharge les deux fichiers du Dictionnaire de Nomenclature `main` et `amas` en utilisant l'URL trouvée dans le paramètre urlmainfile. Elle envoie également une requête à SIMBAD pour récupérer tous les noms d'identificateurs commençant par “NAME ” ou “p. ”

Pour chacune de ces trois étapes, les sorties standard et erreur sont redirigées vers des fichiers du répertoire `nomenclature` qui sont ensuite relus pour afficher le résultat de l'analyse des formats dans la fenêtre principale.

Pour le fichier `main`, les exemples ne correspondant pas aux formats sont mis en valeur en rouge dans le fichier `main_bad_ex.htm` accessible en cliquant sur le nombre de ces exemples.

La date du dictionnaire (récupérée dans la page du dictionnaire[6] dont le début de l'URL provient du paramètre urlmainfile) est alors enregistrée dans la préférence dicoDate pour que DJIN puisse tester à chaque démarrage si une nouvelle version du dictionnaire est disponible.

La phase d'initialisation de DJIN est ensuite recommencée pour charger le nouveau contenu du fichier `liste [46]` .

L'option *`-dico`* ordonne à DJIN de regénérer le fichier `liste [46]` puis de quitter. Elle est utilisée lors de la création d'une nouvelle version.

L'option *`-dicoUrl`* regénère également le fichier `liste [46]` mais permet d'indiquer une URL pour l'emplacement de `main` et `amas`. Elle ne fait pas quitter le programme et inclut en plus l'analyse des exemples NED par rapport à leur format. Elle est utilisée par la personne en charge des modifications du dictionnaire (Marianne Brouty en 2010) pour analyser une future version du dictionnaire.

2.6. Aide

2.6.1. Documentation utilisateur

La commande *Help, Help* affiche dans un navigateur le contenu du fichier `doc.htm` qui contient la documentation utilisateur de DJIN.

Ce fichier a été à l'origine généré à partir de `~/objectname/djin/docbook/ObjectNameRecognitionFR.htx` au moyen du programme `cgiprint`. Il a depuis été converti au format Docbook en `~/objectname/djin/docbook/ObjectNameRecognitionFR.xml`. Il existe aussi en version anglaise dans le ficher `docEnHtm`

2.6.2. A propos

En plus des informations habituelles, la commande *Help, About* donne :

- la date de création de la version actuelle (c'est la date du fichier `objectname.jar`)
- Le nombre d'expressions régulières de l'arbre principal
- La liste des modules externes utilisés par DJIN:
  - PDFBox[7] (`PDFBox-0.7.3.jar` et `FontBox-0.1.0.jar`)
  - JPedal[8] (`jpedalSTD.jar`)
  - JSAMP[9] (`jsamp-1.0.jar`)
  - Weka[10] (`weka.jar`)

2.6.3. Mise à jour de DJIN

Il est possible d'installer la dernière version de DJIN par une commande du menu. Celle-ci vérifie auparavant qu'il n'y a pas d'état courant sauvegardé.

La commande *Help, Upgrade* lance le script `onrinstall.sh` avec l'option *`-version`* ce qui copie le fichier `version.txt` depuis le serveur SIMBAD sur le poste utilisateur sous le nom `available_version.txt` . Le numéro de version qu'il contient est comparé avec celui de l'installation courante par la fonction `Menu.isNewDjinVersionAvailable` et la mise à jour est

[6] http://vizier.u-strasbg.fr/cgi-bin/Dic

proposée à l'utilisateur. Se celui-ci accepte, le même script est relancé avec l'option `-f`, ce qui copie le paquet complet `objname.tar.gz` et le décompresse.

2.7. Autres

2.7.1. Démarrage du programme

Actions réalisées au lancement de DJIN :

La fonction `Menu.createAndShowGUI` commence par initialiser l'arbre de recherche en chargeant le fichier `liste [46]` des expressions régulières. La barre de progression rend compte de l'avancement dans ce chargement et affiche “Initialization” (méthode `initSearch`).

L'argument `-edit` est pris en compte pour choisir un journal dans la liste déroulante 

.

L'argument `-volume` est recopié dans la zone 

.

Si un nom de fichier est présent comme premier argument, il est ouvert en tant que fichier PDF

Sinon, si l'argument -bib est présent, le document correspondant est téléchargé et traité

Sinon, si -list est présent, la liste de documents est traitée.

Sinon, si un état courant a été sauvegardé, celui-ci est restauré.

Sinon, si une nouvelle version du dictionnaire est disponible et si l'utilisateur confirme, une nouvelle liste d'expressions régulières est générée (la date du dernier dictionnaire est déterminée sur la page du dictionnaire par la fonction `Menu.isNewMainFileAvailable`).

Sinon, si une nouvelle version de DJIN est disponible, un message s'affiche pour proposer d'utiliser la fonction *Configuration, Upgrade...* .

3. Utilitaires liés à DJIN

Le *classpath* complet pour faire tourner les programmes Java en ligne de commande est :

```
jsamp-1.0.jar:Aclient.jar:weka.jar:bcprov-jdk14-132.jar:bcmail-
jdk14-132.jar:wsdl4j-1.5.1.jar:saaj.jar:commons-
discovery-0.2.jar:commons-
logging-1.0.4.jar:axis.jar:jaxrpc.jar:WS.jar:jpedalSTD.jar:FontBox-0.1.0.jar:PD
```

Comme il est dans le manifeste de l'archive Java `objectname.jar`, il n'y a besoin que de préciser ce fichier.

L'argument pour la machine virtuelle Java *-Xmx512m* permet d'être sûr de pouvoir traiter les gros fichier PDF.

3.1. Extension SAMPDocument pour Firefox

Sur certains sites d'éditeur comme celui de MNRAS[11] le contenu HTML des articles est consultable depuis un navigateur, mais pas n'est pas téléchargeable par DJIN. Il semble qu'un échange de cookies lancé par du Javascript après une redirection en soit la cause.

Un moyen de contourner le problème est de naviguer avec Firefox jusqu'au contenu de l'article puis d'envoyer la page obtenue à DJIN pour qu'il puisse la traiter de la même manière que les autres documents HTML.

[11] http://www.wiley.com/bw/journal.asp?ref=0035-8711&site=1

L'extension pour Firefox SAMPDocument permet donc d'enregistrer en local la page affichée dans le navigateur et d'envoyer un message SAMP à DJIN contenant le chemin du fichier HTML à ouvrir.

Pour installer l'extension, il faut ouvrir le fichier `sampDocument.xpi` avec Firefox puis choisir l'emplacement du répertoire temporaire où seront enregistrés les articles en cliquant sur le bouton *`Preference`* de la ligne de SAMPDocument dans la fenêtre des *Modules complémentaires* de Firefox.

Les actions de l'extension sont programmées en Javascript dans le fichier source `djin/firefox/samp/chrome/content/djin.js`.

Les autres fichiers source du même répertoire :

- `status_overlay.xul` modifie l'interface de Firefox pour ajouter la commande *Outils, Send to DJIN* dans le menu
- `djinPref.xul` contient la description de la fenêtre de préférences de l'extension
- `djinPref.js` contient le code pour faire fonctionner cette fenêtre
- `about.xul` contient la description de la fenêtre *`A propos`* de l'extension.
- `nsXmlRpcClient.js` contient le module XML-RPC client utilisé pour envoyer les messages SAMP

L'envoi des messages se fait en utilisant le protocole XML-RPC[12], qui était inclus dans Firefox 2.0 mais plus dans Firefox 3.0 ce qui a obligé l'ajout du dernier fichier. Même ainsi, l'extension ne fonctionne pas sur certains postes car les modules dont disposent Firefox (à ne pas confondre avec les extensions ni les plugins) peuvent être très différents d'une installation à l'autre.

Pour qu'un programme puisse recevoir des message SAMP, il faut qu'un hub SAMP tourne sur la machine. DJIN contient un tel hub (fonction `SampManager.startHub`) qui peut être lancé au démarrage en cochant la préférence samphub. La fenêtre des préférences de DJIN permet également d'afficher un moniteur SAMP qui affiche la liste de tous les programmes implémentant le protocole SAMP qui tourne sur le poste.

Note

L'atlas du ciel Aladin[13] fonctionne sur le même principe.

Une autre manière de faire tourner un hub SAMP est de lancer celui contenu dans l'archive Java `jsamp-1.0.jar` de la manière suivante :

`java -cp jsamp-1.0.jar org.astrogrid.samp.xmlrpc.HubRunner` (Ce programme affiche en même temps un moniteur SAMP).

La fonction `SampMessageHandler.processCall` ordonne à DJIN l'ouverture et le traitement du fichier HTML dont l'URL est dans le message reçu.

3.2. Comparaison de symboles graphiques

Il est possible de visualiser la comparaison de deux symboles graphiques par DJIN.

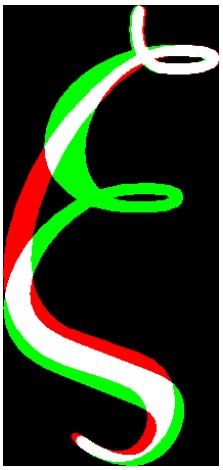
La classe de comparaison d'images de symboles graphiques utilisée par DJIN lors de l'extraction de texte depuis un document PDF contient une fonction `GraphicSymbol.main` qui crée une image montrant la superposition de deux caractères choisis parmi les images de symboles en blanc sur fond noir enregistrés dans le répertoire symbols.

L'image résultat montre les deux caractères en vert et rouge avec la partie commune en blanc.

[12] http://www-archive.mozilla.org/projects/xmlrpc/
[13] http://aladin.u-strasbg.fr/java/doctech/cds/aladin/SAMPManager.html

Le pourcentage de pixels différents permet de décider de la valeur du paramètre d'extraction ratiosymbol (entre 0 et 1).

Figure 21. Visualisation de la comparaison de deux symboles graphiques



La syntaxe est la suivante :

```
java -cp objectname.jar cds.ObjectName.Extraction.GraphicSymbol image1 image2 [-result file]
```

où *image1* et *image2* sont les deux images source et *file* est le chemin du fichier résultat.

Exemple de résultat :

```
First  picture dimension : 213x451 = 96063 pixels
Second picture dimension : 211x451 = 95161 pixels
Identical pixels within the first picture frame : 80946
Non black pixels out of the first picture frame : 0
Percentage of different pixels : 18
```

3.3. Fréquences des acronymes choisis pour ceux détectés

Pour pouvoir proposer d'abord les noms d'objet les plus probables lors d'un ajout dans la liste des identificateurs, le fichier `acro_freq` contient des données permettant de comparer entre eux les acronymes du dictionnaire. La fonction `AcroFreq.main` permet de remplir ce fichier.

La fonction `AcroFreq.readAndWrite` récupère sur l'entrée standard le résultat d'une requête à SIMBAD du genre :

```
select ref_raw_id, ref_norm_id from has_bib_ref h , bib_ref b where oidbib=oidbibref and bibcode>'2009' and ref_raw_id is not null and ref_norm_id is not null
```

Les champs `ref_raw_id` et `ref_norm_id` sont séparés par le caractère |

Les acronymes des noms d'objet sont déterminés par `ObjName.acronymFromString` et la sortie est constituée de lignes au format suivant :

```
raw id acronym ; norm id acronym ; nb
```

Le dernier champ est donc le nombre de fois où un acronyme résultat a été la bonne réponse pour un acronyme de raw id donné sur la période couverte par la requête.

3.4. Visualisation des informations sur les caractères d'un fichier PDF

Le programme `PDFEdit` est un petit visualiseur de document PDF utilisant les fonctions de PDFBox et affichant les coordonnées des caractères d'un document PDF.

Ces coordonnées ainsi que leur hauteur sont utiles pour décider de la valeur des paramètres d'extraction mais l'utilisation du paramètre show_height donne aussi ces informations en utilisant seulement DJIN.

Une fois le programme lancé, cliquer sur *Open* pour ouvrir un fichier. Les coordonnées des groupes de caractères s'affichent dans la barre de titre un cadre rouge met la position en valeur quand on passe le pointeur de la souris au-dessus. Un clic sur une position affiche des informations supplémentaires comme les chiffres de la matrice de transformation des caractères dans le document. Cette matrice est utilisé pour pencher les caractères, les tourner et les modifier en taille.

La modification de ces données ne fonctionne pas.

3.5. Tests unitaires et d'intégration

Les tests suivants sont programmés pour être lancés automatiquement et vérifier le plus grand nombre de fonctionnalités possibles. Ils sont lancés à chaque création d'une nouvelle version de DJIN qui s'interrompt dès qu'une erreur est détectée.

Ils utilisent le module JUnit 3.8.1 qui exécute toutes les fonctions dont le nom commence par “test” dans les classes dérivées de `TestCase` en utilisant le mécanisme de Reflection[14].

Les tests unitaires vérifient des fonctions de base et les tests d'intégration des fonctionnalités complètes. L'interface graphique n'est pas testée.

Les données de test utilisées sont dans le sous-répertoire `junit/test` des sources archivés par Subversion. Les sources doivent être extrait dans un répertoire nommé `objectname` situé dans le HOME du développeur.

- `TestBase.testRegularExpression` lance la génération des expressions régulières à partir des fichiers du dictionnaire `main` et `amas` récupérés sur la machine VizieR. Le nombre des expressions trouvées ne doit pas baisser de plus de 5% par rapport au dernier test.
- `TestBase.testExtractTextFromPDF` extrait le texte d'un document PDF et effectue des vérifications sur le résultat
- `TestBase.testExtractTextFromHTML` fait de même avec un document HTML.
- `TestBase.testSearchNames` extrait le texte d'un document PDF puis recherche les noms d'objet et vérifie la présence de l'un d'eux ainsi que son nombre d'occurrences et les positions de celles-ci.
- `TestBase.testReadOneParfile` extrait d'un fichier parfile les bibcodes et titres puis les écrit dans un tableau HTML, relit ce document HTML en utilisant le programme YesNo et vérifie le nombre de noms d'objet trouvés dans les titres.
- `TestRobot.testMainAandA` ouvre des articles de A&A en PDF dont les noms de fichier sont contenus dans le fichier `~/objectname/djin/junit/test/aa.txt` et vérifie en utilisant le programme Robot pour chacun que les noms d'objet du fichier résultat `~/objectname/djin/junit/test/aaresult.csv` ont bien été détectés.
- Les fonctions `testMainApJ, testMainAJ, testMainMNRAS` font de même avec les listes d'articles d'ApJ, AJ et MNRAS.

[14] http://ricky81.developpez.com/tutoriel/java/api/reflection/

3.6. Programmes intégrés à DJIN

Ces programmes, à l'origine indépendants et en ligne de commande, ont été intégrés à DJIN et sont accessibles par les menus.

3.6.1. Génération des expressions régulières

Génération des expressions régulières à partir d'un fichier du dictionnaire (`main` ou `amas`). La classe est maintenant manipulée par `RegularFactory` lors du chargement des formats du dictionnaire.

La syntaxe est la suivante :

`java -cp objectname.jar cds.ObjectName.Recognition.Regular` *`[OPTIONS] input [output]`*

où :

- *`input`* est le chemin du fichier du dictionnaire à lire
- *`output`* est le fichier de sortie où sont écrites les expressions régulières [46]. S'il est absent, elles vont sur la sortie standard.
- Options :
  - *`-error`* : fichier pour les erreurs et les exceptions non traitées
  - *`-result`* : fichier pour la description des résultats (affichage des totaux et autres statistiques ainsi que la liste des exemples ne correspondant pas à leur format).
  - *`-reserved`* : fichier contenant une liste de mots réservés. Ces mots étaient repérés dans les formats et n'étaient pas transformés. Ce système n'est plus utilisé depuis que l'usage des accolades dans les formats est systématique.
  - *`-remove`* : fichier contenant une liste d'expressions régulières devant être retirées du résultat parce qu'elles génèrent trop de bruit dans les détections de noms d'objet. Ce système n'est plus utilisé car les formats posant ce problème ont été retirés du dictionnaire.
  - *`-greek`* : fichier contenant les noms des lettres grecques [46]
  - *`-strategy`* : numéro de stratégie utilisée pour obtenir une expression régulière à partir du format et des exemples. Les seules à être encore utilisées sont : 3 (multi-exemple) et 4 (fichier des amas sans exemples)
    - 0 : Les formats sont transformés sans tenir compte des exemples
    - 1 : Les caractères du début du format ne sont pas transformés s'il existe un exemple où ils apparaissent non transformés.
    - 2 : Les formats sont transformés lettre par lettre en partant de la fin. Le résultat qui satisfait le plus d'exemples est conservé.
    - 3 : Toutes les combinaisons de transformation des éléments du format étaient testées pour optimiser le nombre d'exemples satisfaits. Depuis que les accolades sont utilisées systématiquement dans les formats, tous les caractères hors accolades sont systématiquement transformés. Les exemples non satisfaits génèrent un message d'avertissement.
    - 4 : Le fichier des amas ne contient pas d'exemples, les acronymes SIMBAD et principal y sont gérés différemment.

3.6.2. Annotation de document PDF

Ce programme permet d'annoter un document PDF avec une expression régulière ou une liste d'expressions contenue dans un fichier. La syntaxe est :

```
java -cp objectname.jar cds.ObjectName.Highlight.Annot [OPTIONS] PDF_file PDF_output_file
```

Paramètres :

- *PDF_file* Document PDF lu
- *PDF_ouput_file* chemin du document PDF annoté
- Options :
  - *-password* : Mot de passe pour les documents cryptés
  - *-start* : Numéro de la première page à traiter (basé sur 1)
  - *-end* : Numéro de la dernière page à traiter
  - *-edit* : Nom du journal (Ex : AJ, ApJ, MNRAS)
  - *-volume* : Numéro de volume
  - *-config* : Chemin du fichier de configuration XML
  - *-searchFile* : Chemin d'un fichier contenant une expression régulière par ligne
  - *-search* : Expression régulière

Un des deux derniers paramètres est obligatoire pour indiquer le ou les formats des mots à annoter.

3.6.3. Extraction de texte

Extraction du texte d'un document PDF. La syntaxe est la suivante :

```
java -cp objectname.jar cds.ObjectName.Extraction.Extract [OPTIONS] PDF_file [Text File]
```

La signification des paramètres *PDF_file, -password, -start, -end, -edit, -volume, -config* est la même que pour le programme d'annotation.

L'argument *-console* indique d'afficher le texte extrait sur la sortie standard. Dans le cas contraire, l'argument *Text File* est utilisé.

-encoding permet d'indiquer un encodage pour la sortie

3.6.4. Traitement de listes de documents en ligne de commande

Pour les besoins des tests et de la validation, il est possible de lancer en ligne de commande le traitement de listes de documents. La syntaxe est la suivante :

```
java -cp objectname.jar cds.ObjectName.Recognition.Robot (-list file | -abstractDir dir | -version) [OPTIONS] [outputDir]
```

Pour les tests, lorsque l'option *-list* est utilisée, le programme s'attend à une liste de lignes au format suivant :

`bibcode ; path` où le deuxième champ peut contenir une URL ou un chemin relatif (les chemins absolus peuvent être indiqués avec la syntaxe d'URL `file://...`) ou être vide. S'il est vide le programme tente de récupérer le contenu de l'article à partir du bibcode.

Les noms d'objet sont recherchés dans les textes extraits des documents et leur liste est enregistrée dans le fichier résultat dont le chemin est dans le paramètre d'extraction exportfile. La liste des noms détectés est comparée avec celle donnée par SIMBAD et les noms des objets non trouvés vont aussi dans le fichier résultat.

L'option *`-abstractDir`* peut être utilisée à la place de *`-list`* pour le traitement des fichiers parfile.

> **Note**
> La fonction `Robot.main` peut aussi être utilisée pour obtenir le numéro de version de DJIN avec le paramètre *`-version`*.
>
> Commande : `java -cp objectname.jar cds.ObjectName.Recognition.Robot -version`

3.7. Scripts

3.7.1. Installation

La première fois, le programme s'installe à partir d'une archive située sur la machine `simbad`. Les mises à jour peuvent ensuite se faire au moyen du menu *Help, Upgrade* ou en lançant dans un terminal le script `~/.simbad/objectname/onrinstall.sh`

Pour la première installation :

1. Créer le répertoire de DJIN avec la commande suivante :

   ```
   mkdir -p ~/.simbad/objectname
   ```

2. Copier le script d'installation :

   ```
   scp majuser@simbad.u-strasbg.fr:DJIN/onrinstall.sh ~/.simbad/objectname
   ```

3. Lancer le script d'installation :

   ```
   ~/.simbad/objectname/onrinstall.sh
   ```

Ce script d'installation, après avoir vérifié qu'il n'y a pas d'état courant, et demandé confirmation à l'utilisateur copie depuis la machine `simbad` l'archive `objname.tar.gz` et la décompresse.

La date du fichier `objectname.jar` est ensuite écrite dans le fichier `date_maj_XXXX` (où `XXXX` est le nom de l'utilisateur) qui est copié dans `majuser@simbad:DJIN` pour pouvoir facilement dresser la liste des utilisateurs de DJIN avec leur date de dernière mise à jour.

Le fichier `log.txt` est également copié sur la machine `simbad` pour permettre des statistiques sur l'utilisation de DJIN.

Le script peut être lancé avec l'option *`-version`*, ce qui a pour effet de récupérer le fichier `available_version.txt` depuis la machine `simbad` pour connaître le numéro de la dernière version disponible. C'est ce qu'utilise DJIN lorsque l'utilisateur lance *Help, Upgrade*.

L'option *`-f`* permet de lancer l'installation sans demander confirmation à l'utilisateur. C'est ce qu'utilise DJIN après confirmation dans l'IHM.

Sous Unix, un raccourci vers DJIN doit lancer le script `autoobj.sh` qui commence par se placer dans le répertoire d'installation de DJIN afin que le programme puisse trouver son fichier de configuration et ses sous-répertoires.

Sous Windows, le comportement est différent puisque :

- Le répertoire d'installation `objectname` est sur le bureau.
- Le script de mise à jour s'appelle `onrinstall.bat` et est aussi sur le bureau.
- La présence d'un état courant n'est pas vérifiée, seul un message d'avertissement est affiché.
- Les programmes `pscp.exe` et `7z.exe` sont supposés être installés pour permettre respectivement la copie à travers le réseau et la décompression de l'archive.
- Le contenu du répertoire d'installation est sauvegardé en `objectname_old` sur le bureau
- Les fichiers `maj2.bat` et le répertoire `script` sont récupérés de l'ancienne installation.
- DJIN ne peut pas se mettre à jour depuis le menu car sous Windows, on ne peut pas remplacer un fichier en cours d'exécution.

3.7.2. Lancement des programmes

Voici la liste des scripts lanceurs des différents programmes :

- `autoobj.sh` permet de lancer DJIN depuis n'importe quel répertoire
- `objname.sh` permet de lancer DJIN depuis son répertoire d'installation
- `notesToHtml.sh` lance le programme de récupération des notes sur les objets astronomiques qui les enregistre dans un tableau en HTML en y détectant les noms d'objet.
- `onrinstall.sh` est le script d'installation et de mise à jour
- `parfileReader.sh` lance le traitement des listes de document ou la lecture de fichiers parfile selon les options
- simref.sh lance le programme de détection des noms d'objet dans les titres du jour
- vector.sh lance DJIN avec le fichier de configuration du sous-répertoire `vector` qui pointe vers la liste d'expressions régulières de mots-clefs pour la recherche de mots-clefs à la place des noms d'objet
- yesno.sh lance le programme YesNo de détection des noms d'objet dans les titres

3.7.3. Liste des utilisateurs de DJIN

Le script `~/objectname/djin/script/users/listusers.pl` permet d'obtenir la liste des utilisateurs de DJIN avec leur date de dernière mise à jour et celle de leur version installée.

Ce script est archivé dans les sources de Subversion. Il copie en local les fichiers `date_...` depuis `majuser@simbad:DJIN` puis lit ces fichiers pour afficher la liste des utilisateurs classés par date d'installation.

Exemple d'exécution au 20/01/2010 :

```
Program ; Install ; User
2009-11-09 2010-01-18 vonflie
2009-11-09 2009-12-18 aschrey
2009-11-09 2009-11-20 chassagn
2009-11-09 2009-11-20 marianne
2009-11-09 2009-11-13 neuville
2009-11-09 2009-11-10 oberto
2009-11-09 2009-11-10 claude
```

```
2009-11-09 2009-11-09 bonnin
2009-10-08 2009-10-08 bibliocds
2009-10-02 2009-10-02 soizick
2009-05-06 2009-05-07 dubois
2009-03-20 2009-03-20 sborde
2009-02-25 2009-02-27 seb
```

3.7.4. Compilation et création d'une nouvelle version de DJIN

Les actions de compilation des programmes, lancement des campagnes de test automatiques, création des archives (jar, zip, .tar.gz), documentations automatiques, mise en place d'une nouvelle version sont automatisées dans le fichier Ant `~/objectname/djin/djin.xml`.

Ant[15] est un équivalent de Make bien adapté aux programmes en Java (mais pas seulement) dont les fichiers de description de tâches sont en XML.

> **Tip**
>
> Le fichier XSL `antTarget.xsl` situé dans le même répertoire permet d'afficher les tâches sous forme de tableau HTML en ouvrant simplement le fichier XML dans un navigateur.

Le répertoire de destination pour l'installation “maître” de DJIN qui sera prise pour modèle lors de la création du paquet d'installation est paramétré dans le fichier `~/objectname/djin/djin.xml` à : `~/export`

Les tâches Ant peuvent être exécutées par Eclipse ou lancées en ligne de commande à partir du répertoire `~/objectname/djin` avec la syntaxe:

`ant -f djin.xml` *`tâche`*

Les tâches importantes sont :

- *`archive`* compile les sources de DJIN et de simbadned et crée les archives Java `objectname.jar` et `simbadned.jar` ainsi que l'archive Zip de l'extension Firefox `sampDocument.xpi`
- *`test`* lance une campagne de test (durée : quelques minutes)
- *`package`* compile, teste et crée tout le paquet d'installation qui est copié sur la machine `simbad`. Un échec dans les tests interromp aussi la création du paquet.

  Si tout s'est bien passé, la finalisation de la création du paquet d'installation est faite par le script `~/objectname/djin/script/package.sh`
- *`doc`* génère la documentation automatique des sources Javadoc et Doxygen
- *`docbook`* génère le rapport complet (PDF et HTML) à partir du fichier Docbook

4. Rôle de chaque fichier de l'installation

4.1. Répertoire cookie

cookie

Ce répertoire contient les cookies stockés par DJIN lorsqu'il se comporte comme un navigateur pour permettre à l'utilisateur de cliquer sur les liens vers les documents PDF. Chaque fichier contient tous les cookies pour un domaine donnés qui est le nom du fichier. Chaque ligne de ce fichier est un cookie.

4.2. Répertoire current

current

[15] http://ant.apache.org/

Ce répertoire contient la sauvegarde de l'état courant du programme.

- annotated.pdf : Document PDF avec annotations et marque-pages.
- document.html : Document HTML avec annotations.
- extracted : Résultat de l'extraction (texte extrait, positions, listes de noms d'objets, ...)
- identifiers : Liste des identificateurs
- miscellaneous : Informations sur les fenêtres (position dans le texte extrait, zones de saisie : éditeur, numéro de volume, ...)

4.3. Répertoire nomenclature

nomenclature

Ce répertoire contient les fichiers intervenant dans la création de la liste des expressions régulières (menu Configuration, commande Format).

- forms : fichier des formes plurielles. Chaque ligne contient deux champs séparés par un ';'. Le 1er contient la forme de départ et le 2nd la lettre à ajouter pour obtenir le pluriel ou bien le mot complet de la 2ème forme. Pour les acronymes du fichier dictionnaire main, la lettre est ajoutée de manière facultative à l'expression régulière. Pour les noms usuels commençant par NAME ..., une autre expression régulière est ajoutée avec la deuxième forme.
- removed : expressions régulières à supprimer. Les expressions régulières de ce fichier seront supprimées de la liste des expressions régulières générée.
- constellations : liste des constellations. Elles servent à créer une alternative d'expression régulière "(And|Ant|Aps|...)" qui recherche les noms de constellations pour les formats contenant "CCC" (les noms courts doivent être placés après, sinon les noms longs ne seront pas détectés).
- multiple_stars : débuts de noms d'étoiles multiples. Elles ont pour format : "** AAA NNNNAA" mails les auteurs n'écrivent que très rarement "**".

Les fichiers suivants sont générés automatiquement lors du traitement d'une nouvelle version du dictionnaire et affichés dans le résultat de cette transformation.

- errAmas.txt : Erreurs rencontrées lors de la transformation des formats du fichier des amas en expressions régulières
- errName.txt : Erreurs rencontrées lors de la transformation des noms en NAME... de SIMBAD en expressions régulières
- errNed.txt : Erreurs rencontrées lors de l'analyse des informations NED (avec l'option -dicoUrl)
- err.txt : Erreurs rencontrées lors de la transformation des formats du fichier main en expressions régulières
- main_bad_ex.htm : Fichier HTML où les exemples ne correspondants pas à leur format ont été coloriés en rouge. Il est affiché lorsqu'on clique sur le lien Not matching examples
- resAmas.txt : Résultats chiffrés de la transformation du fichier des amas
- res.txt : Résultats chiffrés de la transformation du fichier main

4.4. Répertoire symbols

symbols

Ce répertoire contient les fichiers image représentant les caractères graphiques devant être comparés visuellement lors de l'extraction de texte. Il y a un sous-répertoire par éditeur. Les images du sous-répertoire principal sont celles de la configuration générique.

- meaning.txt : fichier des significations des caractères

Chaque ligne contient 2 champs séparés par un ';'. Le 1er contient la signification qui sera insérée dans le texte extrait et le 2nd contient le nom du fichier image représentant le caractère.

- Les autres fichiers contiennent les images de caractère

4.5. Répertoire vector

vector

Ce répertoire contient les fichier de configuration pour le fonctionnement en mode vecteur. Dans ce mode, le programme cherche des mots permettant de classer les articles en fonction de ce dont ils parlent. La sortie, lors de la validation d'une liste, est un fichier vecteur contenant une ligne par bibcode et une colonne par groupe de mot demandé. Le programme entre dans ce mode lorsque le fichier liste contient des '#' dans les champs de l'acronyme SIMBAD (2ème ligne).

- liste : fichier des expressions régulières

Chaque enregistrement de ce fichier contient 3 lignes séparées par des retours chariot. Pour le mode vecteur, la 1ère ligne est le nom du groupe de mot. Ces mots seront les colonnes du vecteur. La 2ème ligne contient un '#'. La 3ème ligne contient l'expression régulière à chercher dans le texte.

- Les autres fichier ont le même rôle que dans le répertoire principal.

4.6. Répertoire xsl

xsl

Contient les fichiers de transformations XSL utilisés pour extraire les contenus des articles à partir d'une page HTML (fonction HTML from bibcode). Les noms de ces fichiers sont paramétrés dans les paramètres d'extraction htmlfilter.

4.7. Répertoire script

script

Contient les sauvegardes des scripts de mise à jour de SIMBAD générés par DJIN. Le but étant de ne pas perdre le travail des documentalistes en cas de plantage du programme. Les fichiers sont numérotés et enregistrés de manière bouclante (classer par date pour connaître le dernier). DJIN peut ouvrir un script existant à partir de la liste des identificateurs avec le bouton Open script (la liste doit être vide pour que le bouton apparaisse).

4.8. Répertoire principal

Le but est de pouvoir classer comme du bruit les codes postaux contenus dans les lignes d'adresse. Plus une ligne contient de mots correspondant aux expressions régulières de ce fichier, plus il y a de chance pour que ce soit une ligne d'adresse.

- accented_ascii : correspondance entre les caractères accentués et leurs caractères ASCII de base. Utilisé par la création d'un bibcode depuis le presse-papier

- Aclient.jar : Archive Java contenant la classe Aclient permettant d'interroger VizieR (utilitaire gsc4sim)

- acro_freq : Liste d'acronymes de raw id et d'acronymes de noms d'objet correspondant avec leur fréquence.

Utilisé pour classer les propositions de noms d'identificateurs issues du dictionnaire en fonction de leur fréquence de choix par les utilisateurs. Il est généré par le programme cds.ObjectName.Highlight. `AcroFreq`

- address_word : fichier des expressions régulières à chercher pour reconnaître une ligne d'adresse.
-

  autoobj.sh : lancement du programme depuis un répertoire quelconque

  Ce script se place dans le répertoire du programme et exécute son script de lancement. Il est placé dans le répertoire ~./simbad lors de l'installation.

- available_version.txt : c'est la copie du fichier `version.txt` qui se trouve sur le serveur SIMBAD. Cette copie est faite lorsque l'utilisateur demande une mise à jour de DJIN.
- axis.jar : Archive Java contenant le module Apache Axis (utilisé pour se connecter aux Web Services)
- bcmail-jdk14-132.jar : Archive Java du module BouncyCastle[16] (utilisé par le module PDFBox)
- bcprov-jdk14-132.jar : Archive Java du module BouncyCastle[17] (utilisé par le module PDFBox)
- commons-discovery-0.2.jar : Archive Java du module de journalisation Apache utilisé par les Web Services.
- commons-logging-1.0.4.jar : Archive Java d'un module d'Apache utilisé par les Web Services.
- config.xml : Fichier XML de configuration des paramètres d'extraction de texte
- date_maj_<login name> : date de la dernière mise à jour du programme.

  Ce fichier est créé par le script d'installation et copié par celui-ci sur la machine cacao. Il permet de savoir quelle version du programme est installée chez quel utilisateur.

- doc.htm : Documentation du programme (affiché par la commande Help du menu Help)
- docEN.htm : Version anglaise de cette documentation (en ligne sur le wiki eurovotech.org[18])
- FontBox-0.1.0.jar : Archive Java du module FontBox[19] utilisé par le module PDFBox[20]
- greek_letter : Liste des noms des lettres grecques (utilisé par l'extraction de texte et la recherche des noms d'étoiles variables)
- gsc4sim_format : Liste d'expressions régulières correspondant aux formats acceptés par gsc4sim. Pour éviter d'envoyer n'importe quoi à ce programme qui risque de bloquer DJIN
- icone.png : Grande icône du programme. A utiliser comme icône du raccourci pour lancer DJIN.
- iconYesNo.png : icône pouvant être utilisé pour le raccourci du programme YesNoTitleDjin[21] (Détection de noms d'objet dans les titres)
- jaxrpc.jar : Archive Java du module JAX-RPC utilisé par les Web Services (package javax.xml.rpc)
- jpedalSTD.jar : Archive Java du module JPedal[22] utilisé pour obtenir la description graphiques des polices incorporées.
- jsamp-1.0.jar : Archive Java du module JSAMP[23] utilisé pour la communication entre l'extension Firefox et DJIN.
- liste : Liste des expressions régulières et leurs acronymes

Chaque enregistrement de ce fichier contient 3 lignes séparées par des retours chariot. La 1ère ligne contient l'acronyme principal. La 2ème l'acronyme SIMBAD suivi éventuellement d'une tabulation et

de l'acronyme de renvoi (Si elle commence par # le programme génère des vecteurs de mots-clefs). La 3ème l'expression régulière à chercher dans le texte extrait. Elle ne doit pas contenir de caractères '+' ou '*' pour les cardinalités mais seulement des cardinalités fixes comme {n,m} ou '?'.

- listeConst : Liste des expressions régulières pour les constellations

Chaque enregistrement de ce fichier contient 3 lignes séparées par des retours chariot. La 1ère ligne contient le nom abrégé de la constellation précédé du mot 'Constellation', la 2ème le nom complet de la constellation et la 3ème son expression régulière.

- liste_suppr_ap : Liste de mots obligeant à supprimer un nom d'objet si placés après celui-ci.
- liste_suppr_av : Liste de mots obligeant à supprimer un nom d'objet si placés avant celui-ci.
- listeRegExp : Liste d'expressions régulières à ajouter au démarrage.
- listeRegExpHistory : Liste d'expressions régulières à mettre au démarrage dans l'historique des expressions régulières à ajouter (liste déroulante de Search, Add a regular expression), sachant que l'historique de l'utilisateur est lui stocké dans les préférences et ne peut donc pas être effacé par la mise à jour.
- log.txt : Fichier journal.

Une ligne est ajoutée à ce journal à chaque fois qu'un script de commandes de mise à jour SIMBAD est généré ou exécuté. Chaque ligne contient 5 champs séparés par des ';'. Le 1er contient le bibcoce, le 2ème le nombre d'identificateurs validés par l'utilisateur (plusieurs peuvent correspondre au même objet), le 3ème le nombre de noms d'objet différents trouvés dans le texte extrait, le 4ème le nombre de noms d'objets qui ont été ajoutés manuellement par l'utilisateur, le 5ème le nombre de noms d'objets supprimés de l'arborescence.Le but est de pouvoir calculer le pourcentage de bruit dans les noms d'objets détectés par le programme.Ce fichier est copié sur la machine simbad à chaque mise à jour du programme par le script d'installation.

- mainAnnot.html : Fichier principal pour l'affichage de document HTML avec des annotations pour mettre en valeur les noms d'objets en utilisant les Frame HTML
- names.arff : Fichier contenant des données d'apprentissage pour distinguer les vrais noms d'objet des faux positifs
- nice_word : mots d'une phrase pouvant donner à penser que la détection est un vrai nom d'objet.
- notesToHtml.sh : script de lancement du programme de chargement des notes sur les objets SIMBAD pour les afficher en HTML en détectant les noms d'objet (NotesToHtml[24])
- objectname.jar : Archive Java contenant le programme DJIN de reconnaissance des noms d'objet
- objname.bat : Script de lancement du programme pour les machines sous Windows
- objname.sh : Script de lancement du programme pour les machines sous Unix
- objname.tar.gz : Archive contenant tous les fichiers de l'installation
- objnames.csv : Fichier dans lequel les noms d'objet sont stockés lors de la validation d'une liste de document

Une ligne est ajoutée dans ce fichier pour chaque nom d'objet trouvé. Chaque ligne contient 6 champs séparés par des ';'. Le 1er contient le bibcode. Le 2ème contient le nom trouvé. Le 3ème contient le

résultat de la vérification dans SIMBAD : False pour un nom qui n'aurait pas dû être trouvé, NotFound pour un nom qui aurait dû être trouvé, Unchecked lorsque la vérification n'a pu être effectuée, OK pour un nom bien trouvé. Le 4ème est le numéro de version du programme. Le 5ème est l'identificateur principal de l'objet. Le 6ème contient '1' lorsque l'objet existe et '0' sinon.

- onrinstall.bat : Script d'installation du programme sur une machine sous Windows
- onrinstall.sh : Script shell d'installation du programme sur une machine sous Unix
- parameters.txt : Description des paramètres de configuration de l'extraction de texte.

Cette description est affichée dans le cadre jaune de la fenêtre des paramètres de configuration

- parfileReader.sh : Script de lancement du programme en ligne de commande sur une machine sous Unix.

Il permet de valider des listes de bibcodes ou de parcourir des fichiers parfile contenant les titres, résumés et mots-clefs des articles et de générer des fichiers de commande de mise à jour.

- PDFBox-0.7.3.jar : Archive Java du module PDFBox[25]

Ce module permet le chargement des documents PDF, l'extraction du texte, l'ajout d'annotations et de marque-pages

- regexpLarge : Liste d'expressions régulières pour la rechercher élargie (utilisée par la commande Larger search du menu Search)
- saaj.jar : Archive Java du module SAAJ utilisé par les Web Services
- sampDocument.xpi : Archive zip contenant l'extension SAMPDocument[26] pour Firefox. Elle permet de charger un document HTML dans DJIN depuis Firefox.
- simbadned.jar : Archive Java du programme de traduction des identificateurs SIMBAD vers NED et inversement. Ce programme se sert de l'archive de Djin mais pas le contraire.
- simref.sh : Script de lancement du programme YesNoSimref[27] qui récupère les fichier parfile d'un répertoire donné pour y détecter les noms d'objet.
- suspect_word : mots d'une phrase pouvant donner à penser que la détection n'est pas un vrai nom d'objet.
- unicode.html : liste des caractères spéciaux HTML et leur code unicode équivalent. Utilisé par l'annotation des documents en HTML.
- vector.sh : Script shell de lancement du programme en mode vecteur
- version.txt : Numéro de version du paquet d'installation de DJIN
- weka.jar : Archive Java du module d'apprentissage Weka[28]
- wsdl4j-1.5.1.jar : Archive Java du module WSDL4J utilisé par les Web Services
- WS.jar : Archive Java du module de SIMBAD permettant la connection aux Web Services
- yesno.sh : Script de lancement du programme YesNo[29] de détection des noms d'objet dans les titres

4.9. Répertoire temporaire

- annot.pdf : document PDF annoté utilisé par la visualisation de PDF
- annot.html : contenu annoté d'un article en HTML.
- djinhtml.htm : contenu d'un article HTML après transformation par un filtre XSL.

- djinxhtml.xml : page d'un article HTML mise au format XML.
- firefoxContent.htm : page d'un article HTML enregistrée par l'extension SAMPDocument.
- fromUrl.pdf : Contenu PDF enregistré par DJIN après navigation pour repérer un document PDF.
- lstAnnot.html : liste des marque-pages d'un document HTML annoté.
- MainAnnot.html : document maître (déclaration des frames) d'un document HTML annoté.
- names.pdf : liste des noms d'objet détectés dans un document sous forme d'un fichier PDF.
- objname_cmd.txt : fichier de commandes de mises à jour généré par la liste des identificateurs.
- prefs.xml : liste des préférences utilisateurs stockées par DJIN sur un poste.

5. Arguments des lignes de commande

Les programmes utilisent presque tous la même liste d'arguments même si tous ne sont pas traités par chaque programme.

Les arguments de type booléen n'ont pas de valeur.

Les tables d'arguments avec leur valeur sont gérés par des objets dérivés de la classe abstraite Arguments dont les fonctions `tester` et `usage` sont surchargées pour tester la présence des arguments obligatoires à chaque programme et afficher le cas échéant la syntaxe à respecter.

5.1. Liste des arguments

Nom	Type	Utilisation
-abstractDir	Répertoire	Répertoire où chercher les fichiers parfile lors du traitement automatique d'anciens documents
-bib	String	Code bibliographique à traiter
-config	Chemin	Chemin du fichier `config.xml`
-console	Booléen	Résultat du programme sur la sortie standard
-dico	Booléen	DJIN télécharge la dernière version du dictionnaire, génère la liste des expressions régulières et quitte.
-dicoUrl	URL	URL de base où télécharger le fichiers du dictionnaire pour effectuer des vérifications sur les formats SIMBAD et NED.
-download	Booléen	Les documents PDF sont seulement téléchargés (d'après leur bibcode), il n'y pas de traitement.
-edit	String	Nom de journal permettant de choisir un ensemble de paramètres d'extraction
-encoding	String	Encodage pour l'enregistrement du texte extrait dans un fichier par le programme Extract
-end	Entier	Page de fin lors du traitement d'un document PDF ou année de fin pour le programme NotesToHtml
-enteredNames	Booléen	Mise en valeur des noms d'objet présents sur la référence d'après les raw id de SIMBAD lors du traitement d'une liste de document (Robot)
-error	Chemin	Redirection de la sortie erreur pour le programme Regular
-export	Répertoire	Exportation du texte extrait des documents d'une liste (programme Robot) dans le répertoire donné ou dans le répertoire parent de chaque document traité si le contenu est “%p”

Nom	Type	Utilisation
-greek	Chemin	Chemin du fichier des lettres grecques pour le programme Regular (fichier `greek_letter [46]`)
-list	Chemin	Chemin du fichier contenant une liste de documents à traiter où chaque ligne a le format : bibcode;chemin ou URL
-password	String	Mot de passe pour les documents PDF cryptés (programmes Annot, Extract et DJIN)
-remove	Chemin	Chemin du fichier des expressions régulières à rejeter lors de leur génération par Regular (fichier `nomenclature/removed` ou de la liste des expressions régulières de noms d'objet à rejeter (programmes Robot et NotesToHtml)
-result	Chemin	Fichier où écrire le résultat de la superposition de deux symboles graphiques par GraphicSymbol ou la sortie du traitement du dictionnaire par Regular
-search	String	Expression régulière à chercher et mettre en valeur dans un document PDF par Annot
-searchFile	Chemin	Fichier d'expressions régulières (une par ligne) à chercher et mettre en valeur dans un document PDF par Annot
-start	Entier	Page de début lors du traitement d'un document PDF ou année de début pour le programme NotesToHtml
-strategy	Entier	Stratégie d'utilisation des exemples lors du traitement des fichiers du dictionnaire par Regular. Utiliser 3 pour `main` et 4 pour `amas`.
-title	Booléen	Lors du traitement de fichier parfile, les titres extraits sont enregistrés dans des fichiers HTML par Robot
-training	String	Les données d'apprentissage concernant le document en cours de traitement par DJIN sont ajoutées au fichier `names.arff`
-type	Caractère	Type de note SIMBAD (S, L ou I pour : Short, Long, Internal) recherché par le programme NotesToHtml
-version	Booléen	Indique au programme Robot d'afficher le numéro de version et quitter
-volume	Entier	Numéro de volume permettant de choisir un ensemble de paramètres d'extraction
-year	String	Année permettant de composer la requête à SIMBAD pour récupérer les titres à enregistrer dans un fichier parfile par TitleToFile (peut prendre la forme 1987..1990)
1er argument sans nom	Chemin	Fichier d'entrée
2ème argument sans nom	Chemin	Fichier de sortie

6. Correction des problèmes

Liste de problèmes avec leur solution

6.1. L'extraction du texte donne surtout des caractères inconnus ~

Raison : le texte du document est entièrement écrit dans des polices incorporées au fichier PDF.

Solution : modifier le paramètre d'extraction showothersymbol avec la valeur “true” pour ce journal.

6.2. Un symbole graphique n'est pas reconnu et est remplacé par un ~

Solution : Donner une signification à ce caractère pour que DJIN le reconnaisse graphiquement avec la commande du menu *Config, Display unrecognized graphic symbols*

6.3. Un caractère n'est pas reconnu et est remplacé par son code (ex. : \u3B1)

Raison : les caractères dont le code est supérieur à 255 et qui ne sont pas dans la liste des caractères à substituer apparaissent sous cette forme : “\u” suivi du code unicode.

Solution : ajouter dans la liste du paramètre d'extraction codes le code (en hexadécimal précédé de “x”) et la valeur de substitution, de préférence dans la configuration générique ou dans celle du journal si le remplacement doit lui rester spécifique.

6.4. L'ouverture d'un document PDF à partir du bibcode ne fonctionne pas

Raison : Il peut arriver que DJIN ne puisse pas localiser l'URL du contenu de l'article à partir du bibcode parce que le lien fourni par ADS ou le CDS ne fonctionne pas ou parce que le bibcode n'est pas encore dans SIMBAD.

Solution : télécharger le fichier PDF contenant l'article sur le disque local et l'ouvrir par la commande *File, Open PDF File*

6.5. L'ouverture d'un document HTML à partir du bibcode ne fonctionne pas

Raison : identique au cas précédent avec les fichiers PDF.

Solution : la page contenant l'article peut être sauvegardée par le navigateur au format HTML et lue par DJIN avec la commande *File, Open HTML from, File*.

6.6. Aucun texte n'est extrait lors de l'ouverture de l'article en HTML

Raison 1 : à la toute première connection, une information demandée par cookie n'est pas encore disponible en local.

Solution : cliquer une deuxième fois sur le bouton  ou la commande *File, Open HTML from bibcode*.

Raison 2 : la structure du document HTML a changé et ne correspond plus à celle prévue par le filtre XSL.

Solution : modifier le fichier XSL. Son nom est donné par le paramètre d'extraction htmlfilter. Son emplacement est le répertoire `xsl`.

Pour savoir ce qui a changé, consulter le contenu de l'article avec un navigateur et visualiser le code source de la page à l'endroit du titre de l'article par exemple. Exemple de visualisation avec Firefox.

Traitement automatique de listes de documents

1. Recherche de noms d'objet

1.1. Articles complets

DJIN est habituellement utilisé pour traiter un seul article à la fois, mais il peut également traiter à la suite de longues listes de documents, ce qui est utile pour :

- effectuer des tests de validation
- télécharger en local une liste d'articles
- annoter des documents en utilisant les informations de SIMBAD
- collecter des informations supplémentaires (raw id, occurrences, positions) pour les articles déjà traités
- extraire les textes d'une série d'article
- annoter (tagger) les textes extraits
- générer des statistiques sur les mots avant et après les noms d'objets détectés et validés

Ce traitement de liste est accessible dans DJIN par le menu *File, Open a list of bibcodes for validation* ou en ligne de commande.

Avec l'IHM, si un nom de journal est sélectionné, il est utilisé pour tous les articles, sinon, il est déterminé à partir des bibcodes.

Rappel du format du fichier liste :

`bibcode ; path` où le deuxième champ peut contenir une URL ou un chemin relatif (les chemins absolus peuvent être indiqués avec la syntaxe d'URL `file://...`) ou être vide. S'il est vide le programme tente de récupérer le contenu de l'article à partir du bibcode.

Le résultat du traitement de chaque article s'affiche dans la fenêtre principale (ou sur la sortie standard) :

Exemple obtenu avec le fichier `~/objectname/djin/junit/test/mnras.txt` de la campagne de test :

```
2006MNRAS.373..653R  : 18 OK, 7 false names, 3  not found
2006MNRAS.371.1793K  : 24 OK, 3 false names, 509  not found
2005MNRAS.357...12M  : 2 OK, 2 false names, 1  not found
Total : 44 OK, 12 false names, 513  not found (92 % not found, 78 % good names)
```

La colonne *OK* représente les noms d'objet détectés qui doivent effectivement l'être d'après la liste simbo fournie par SIMBAD. Les *false names* sont ceux détectés mais absents de la référence, les *not found* sont ceux qui auraient dû être détectés mais n'ont pas été trouvés.

Le pourcentage *not found* représente les non trouvés par rapport au nombre présent sur la référence. Le pourcentage *good names* représente les objets présents sur la référence par rapport à ceux qui ont été détectés.

Avant le traitement de la liste, la fonction `Validator.validate` lit le fichier résultat dont le nom est dans le paramètre exportfile (en général, c'est `objnames.csv`) de manière à pouvoir y ajouter les résultats (une ligne par nom d'objet trouvé ou non trouvé) de chaque bibcode. Les résultats d'un bibcode déjà présent seront remplacés.

Les paramètres d'extraction (fichier de configuration) suivants permettent de modifier les traitements :

- taggeddir indique un répertoire de destination dans lequel sont enregistrés les textes extraits où les noms d'objet détectés et présents sur la référence dans SIMBAD sont signalés par des balises `\objE{ }`.
- annotateddir indique un répertoire de destination dans lequel les documents annotés sont enregistrés (les couleurs des annotations sont fonction de l'existence dans SIMBAD et de la présence sur la référence)
- masterdir indique un répertoire de destination dans lequel sont enregistrés les originaux des documents (avant annotation).
- statistic indique un nom de fichier où sont écrites des statistiques sur les détections des noms d'objet : pour chaque nom d'objet détecté et présent sur la référence dans SIMBAD, une ligne y est écrite dans laquelle il est délimité par des caractères | avec les cinq mots précédents et suivants dans la ligne du texte extrait. Exemple : `process of falling into |A1308|. However, the radial velocity`

Les arguments de la ligne de commande (celle de DJIN ou du traitement des listes) ont les effets suivants :

- Le dernier argument (sans nom) précise un répertoire de destination dans lequel seront enregistrés les fichiers de commandes de mise à jour pour entrer dans SIMBAD les informations supplémentaires (raw id, occurrences, positions).
- *`-edit`* permet d'imposer un nom de journal et son ensemble de paramètres d'extraction. Sinon, le journal est choisi à chaque ligne d'après le bibcode.
- *`-bib`* permet d'indiquer un des bibcodes de la liste par lequel le traitement doit commencer. Ce qui évite de relancer toute la liste ou de modifier le fichier en cas d'interruption.
- *`-export permet d'indiquer un répertoire de destination dans lequel le texte extrait des documents PDF est exporté au format HTML`*. Le contenu de l'argument peut être “%p” pour indiquer que le résultat de l'extraction doit être enregistré dans le répertoire parent du fichier lu.
- *`-remove`* permet d'indiquer un fichier contenant des expressions régulières correspondant à des noms d'objet à rejeter. Si l'argument est présent, les noms d'objet ne sont pas vérifiés dans SIMBAD
- *`-enteredNames`* indique que seuls les noms d'objet présents sur la référence dans SIMBAD aux bonnes positions sont conservés et annotés dans le document.
- *`-download`* indique de ne rien faire hormis l'enregistrement du document original, ce qui permet de télécharger une série d'articles sans avoir à les traiter (extraction, détection, ...).

Exemples de lignes de commandes possibles depuis le répertoire de DJIN :

- Génération des fichiers de commande pour informations supplémentaires de documents déjà traités `./parfileReader.sh -list ~/objectname/djin/junit/test/mnras.txt` *`répertoire_dest`*
- Récupération de documents PDF à partir de bibcodes (le paramètre masterdir doit être renseigné) :

  ```
  ./objname.sh -edit MNRAS -download -list /tmp/mnras.txt
  ```

- Annotation de documents avec les noms d'objets entrés dans SIMBAD (le paramètre annotateddir doit être renseigné) :

```
./parfileReader.sh -edit A\&A -list /tmp/aa.txt -enteredNames
```

1.2. Fichiers parfile

Pour les articles dont les fichiers PDF ne sont pas disponibles, ou lorsque ces fichiers ne contiennent pas de texte à extraire, DJIN peut travailler sur des fichiers au format parfile. Ces fichiers doivent au moins contenir les bibcodes et les titres et éventuellement les résumés et les mots-clefs.

Ce traitement est obtenu avec la commande `java -cp objectname.jar cds.ObjectName.Robot` (ou le fichier `parfileReader.sh`) et l'option *`-abstractDir`* suivie du répertoire où se trouvent les fichiers à traiter.

La fonction `ParfileReader.read` lit tous les fichiers de ce répertoire et traite les lignes commençant par :

- `%R` pour les bibcodes
- `%T` pour les titres
- `%K` pour les mots-clefs
- `%B` pour les résumés

Un texte extrait est constitué avec ces éléments et les noms d'objet y sont recherchés. Lorsqu'un répertoire de destination est précisé sur la ligne de commande (dernier argument sans nom), des fichiers de commandes de mise à jour de SIMBAD y sont générés avec le nom des fichiers parfile et l'extension `.sim`. Ils contiendront les commandes d'ajout des informations supplémentaires (ex : `o Centaurus A, +tkao, +Centaurus A;NGC 5128, =4`) pour les objets présents sur la référence dans SIMBAD.

Le programme tient compte des noms d'objet qui sont signalés par les balises suivantes dans le fichier parfile

- `\\object{}` : nom d'objet taggé
- `\\objS{}` : nom d'objet de SIMBAD
- `\\objnS{}` ou `\\objNS{}` : nom d'objet hors SIMBAD à ignorer
- `\\objC{,}` ou `\\objM{,}` : nom(s) d'objet corrigé (le raw id est fourni).

Si l'option *`-title`* est présente, des fichiers au format HTML contenant les bibcodes et titres des articles dans un tableau sont générés dans le répertoire de destination.

Cette option est plutôt à utiliser avec l'option *`-remove`* qui indique un chemin de fichier contenant des expressions régulières pour les noms d'objet à rejeter. Dans ce cas, les noms d'objet retenus ne sont pas seulement ceux présents sur la référence dans SIMBAD.

Les tableaux de titres sont répartis dans 3 fichiers :

1. Les titres ne contenant que des noms d'objet présent sur la référence (en vert) sont écrits dans un fichier nommé comme le fichier parfile de départ et finissant par `.1.html`
2. Ceux contenant des noms d'objet existant et absent de la référence (en rouge) sont écrits dans un fichier finissant par `.html`
3. Les autres (noms rejetés et inexistants) (en rose, bleu, violet) sont écrits dans un fichier finissant par `.0.html`

Voici un exemple de ligne de commande qui génére des fichiers de commande pour entrer les informations supplémentaires de documents déjà traités dans SIMBAD (à lancer dans le répertoire de DJIN) :

`./parfileReader.sh -abstractDir` *`répertoire_source répertoire_dest`*

La sortie contiendra une ligne pour chaque bibcode dans les fichier parfile avec le nombre de noms d'objet validés et le nombre de détections de noms absents de la référence avec leur liste entre [crochets]:

```
2008ApJ...686..948B : 13 object names, 1 false names [Magellanic Clouds]
```

Exemple de génération de tableau HTML pour le programme YesNo. Les fichiers parfile de départ peuvent être générés par le programme TitleToFile.

`./parfileReader.sh -abstractDir` *`répertoire_source`* `-title -remove` *`removed_names répertoire_dest`*

Les tableaux pour tous les titres de SIMBAD jusqu'à l'année 2007 incluses ont été générés et sont consultables sur le site interne[1]

Un exemple de fichier *`removed_names`* se trouve dans le répertoire des sources `~/objectname/djin/`.

1.3. Notes de SIMBAD

L'objectif de ce programme était de repérer dans les notes de SIMBAD des noms d'objet non taggés.

Syntaxe (depuis le répertoire de DJIN) :

`./notesToHtml.sh -start` *`start_year`* `-end` *`end_year output_dir`*

La première étape, effectuée par la fonction `NotesToHtml.readNotes`, consiste à envoyer une requête à SIMBAD pour récupérer les contenu des notes.

Cette requête est construite à partir des arguments obligatoires *`-start`* et *`-end`* qui donnent les années de début et de fin des notes chargées. Et de l'argument *`-type`* qui permet de limiter la requête à un type de note en particulier : S, L ou I (pour les notes courtes, longues ou internes).

L'argument obligatoire *`output_dir`* donne le répertoire de destination dans lequel est d'abord écrit le résultat brut de la requête dans un fichier d'extension `.txt`. Si ce fichier est présent, la requête n'est plus relancée.

Les noms d'objet sont ensuite détectés et un fichier HTML est généré qui contient sous forme de tableau les notes dans lesquelles au moins un nom d'objet existant non taggé a été détecté. Un fichier de commandes de mise à jour d'extension `.sim` est également généré pour remplacer le contenu des notes concernés par un contenu où les noms d'objet sont taggés.

L'argument *`-remove`* peut être utilisé pour indiquer un fichier d'expressions régulières correspondant à des noms d'objet à rejeter. Le fichier utilisé est archivé dans Subversion : `~/objectname/djin/remove_from_notes`

Les tableaux HTML générés en 2009 sont consultables sur le site interne[2]

1.4. YesNo : validation des noms dans les titres

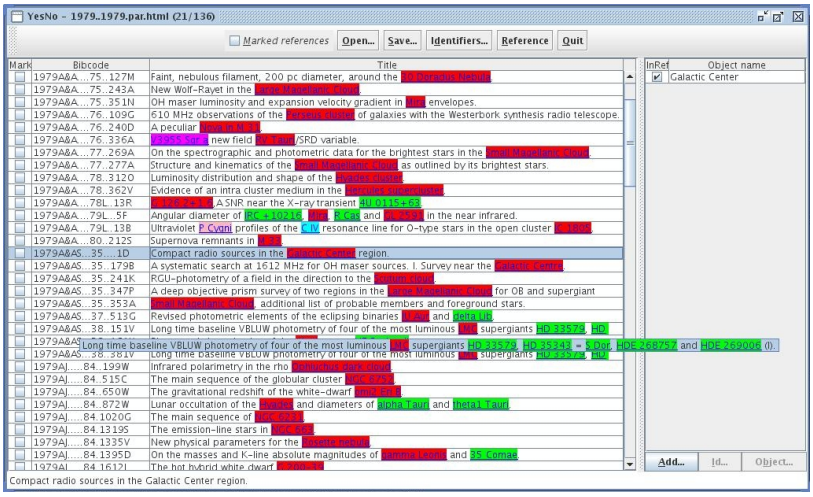
Les tableaux HTML générés par le programme de lecture de fichier parfile contenant les titres des articles dans lesquels les noms d'objet sont mis en valeur peuvent être présentés par le programme YesNo (`TitleDjin`) pour permettre leur validation par un astronome ou documentaliste.

[1] http://snob.u-strasbg.fr/~bonnin/titles.html
[2] http://snob.u-strasbg.fr/~bonnin/notes.html

Le script `yesno.sh [48]` permet de lancer le programme. Si l'argument -config précise un fichier de configuration `config.xml` , SIMBAD peut être interrogé et le bouton *`Identifiers`* apparaît qui permet de gérer la liste des identificateurs comme dans DJIN (A ceci près qu'il y a plusieurs bibcodes).

La première chose à faire est d'ouvrir un fichier HTML contenant un tableau de titres en cliquant sur *`Open`*.

Figure 1. Programme YesNo



La liste principale contient une ligne par article qui affiche le titre, le bibcode et une case permettant de cocher la référence. Il est possible de n'afficher que les références cochées, ou non cochées ou toutes en cliquant sur la case *`Marked references`* à 3 états.

Certains titres sont trop longs pour tenir dans la colonne. Ils sont aussi affichés dans la barre d'état en bas de la fenêtre lorsque la ligne est sélectionnée ou dans la bulle qui apparaît en passant le pointeur de la souris dessus.

Le code des couleurs des noms d'objet est le même que dans DJIN.

Lorsqu'une ligne est sélectionnée, la liste des noms d'objet du titre apparaît dans la partie droite. La case à cocher *`InRef`* indique si le nom doit être présent sur la référence. Si oui une commande l'ajoutera avec son raw id, la position “t” (pour titre) et le nombre d'occurrence 1.

L'identificateur SIMBAD de l'objet apparaît dans la bulle qui s'affiche en passant le pointeur de la souris sur le nom d'objet à droite. Le bouton *`Id...`* permet de modifier cet identificateur.

Le bouton *`Add...`* permet d'ajouter un nom d'objet non détecté de la même manière que dans DJIN. Par défaut, le texte du nom d'objet à ajoute est celui sélectionné dans la barre d'état.

Lorsqu'un nom d'objet est sélectionné, le bouton *`Object`* permet d'afficher sa description par SIMBAD dans un navigateur. De même, lorsqu'un article est sélectionné, le bouton *`Reference`* affiche sa description.

Les scripts de mise à jour peuvent alors être exécutés, mais pour les longs fichiers, il est plus judicieux d'utiliser la mise à jour de SIMBAD.

Chaque opération du programme est automatiquement sauvegardée dans un fichier dont le nom commence par `autoSave` et qui est dans le même répertoire que le fichier ouvert. Ceci afin de ne pas perdre le travail de l'utilisateur.

Les modifications effectuées peuvent être enregistrées par l'utilisateur au moyen du bouton *`Save...`* dans un fichier au même format (tableau en HTML).

Dans ce fichier, la fonction `TitleList.saveHtmlFile` encode les identificateurs dans l'URL de leur lien vers SIMBAD. Les noms d'objet ajoutés par l'utilisateur sont ajoutés à la fin des titres entre [crochets]. Les marques sur les références sont signalées par le commentaire `<!--marked-->`. Le rejet d'un nom d'objet est signalé par le commentaire `<!--deleted-->` à la fin du nom.

Les classes de ce programme sont `TitleDjin` pour la fenêtre principale, `TitleList` pour la liste des titres et `Title` pour chaque titre.

1.5. Simref : validation dans les titres du jour

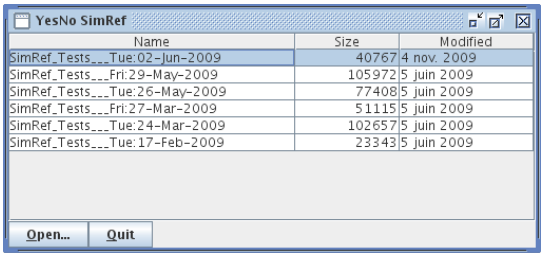
L'objectif de ce programme est de traiter les titres du jour dès qu'ils arrivent. Ceux-ci sont lus dans un fichier parfile et transformés à la volée en tableau HTML. Ensuite, le fonctionnement est le même que pour le programme YesNo.

Pour lancer le programme, la syntaxe est la suivante (depuis le répertoire de DJIN) :

`./simref.sh` *`output_dir`*

Où *`output_dir`* est le répertoire où se trouvent les fichiers parfile à traiter.

Figure 2. Liste des fichiers contenant des titres du jour



Cette fenêtre liste les fichiers du répertoire qui n'ont pas pour extension `.html` ou `.sim`. On peut classer les fichiers en cliquant sur les entêtes de colonne. L'ouverture se fait par le bouton *`Open`* ou un double-clic.

A l'ouverture, les champs %R et %T sont lus et les références et titres sont écrits dans un tableau HTML dont l'extension est `.html`. Si ce fichier existe déjà, il est simplement ouvert.

Le programme YesNo ouvre alors le fichier HTML. A ce stade, les noms d'objets ne sont pas encore vérifiés dans SIMBAD et leur couleur est orange. Cette vérification commence dès l'affichage de la liste de titres.

La liste simbo de la référence est également demandée pour pouvoir afficher en vert les noms d'objet déjà entrés.

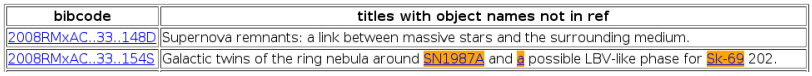
Lorsqu'un tableau HTML existe déjà pour un fichier demandé, il est ouvert directement. Les noms d'objet sont déjà vérifiés et les modifications de l'utilisateur sont conservées.

Exemple de fichier parfile lu :

```
================ Detail of New References =================
%#Parfile(5.23) on: simref.0(17 Feb 2009 23:28:02)
%#Execution date: 17 Feb 2009 23:29:07

%R 2008RMxAC..33..148D
%A DUBNER G.
%J Rev. Mex. Astron. Astrofis. Serie de Conf., 33, 148-153 (2008)
%T Supernova remnants: a link between massive stars and the surrounding medium.
```

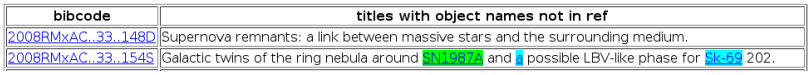
```
%O=17
%D 17.02.09 17.02.09

%R 2008RMxAC..33..154S
%A SMITH N.
%J Rev. Mex. Astron. Astrofis. Serie de Conf., 33, 154-156 (2008)
%T Galactic twins of the ring nebula around SN1987A and a possible LBV-like
   phase for Sk-69 202.
%O=10
%D 17.02.09 17.02.09
```

Figure 3. Transformation du fichier parfile en fichier HTML



bibcode	titles with object names not in ref
2008RMxAC..33..148D	Supernova remnants: a link between massive stars and the surrounding medium.
2008RMxAC..33..154S	Galactic twins of the ring nebula around SN1987A and a possible LBV-like phase for Sk-69 202.

Figure 4. Fichier HTML après vérification des noms d'objet



bibcode	titles with object names not in ref
2008RMxAC..33..148D	Supernova remnants: a link between massive stars and the surrounding medium.
2008RMxAC..33..154S	Galactic twins of the ring nebula around SN1987A and a possible LBV-like phase for Sk-69 202.

2. Recherche de mots-clefs avec génération de vecteurs

DJIN et les outils en ligne de commande peuvent également chercher des mots-clefs à la place des noms d'objet. Ce qui peut servir à indexer des listes de documents par rapport à des groupes de mots-clefs pour ensuite établir des cartes de Kohonen[3] ou alimenter un moteur de recherche.

Chaque vecteur est une ligne correspondant à une référence et les colonnes représenntent les groupes de mots-clefs. Ces colonnes sont remplies avec les nombres de mots du groupe détectés.

2.1. Calcul des nombres d'occurrences par groupe de mots-clefs

Pour obtenir les nombres d'occurrences par groupes de mots-clefs d'une liste de documents, il suffit d'indiquer à DJIN un fichier de configuration qui pointe (paramètre astrodir) vers un répertoire contenant un fichier d'expressions régulières [46] correspondant aux mots-clefs et de lancer le traitement d'une liste de documents.

Le sous-répertoire `vector` contient les fichiers de configuration et d'expressions régulières pour cette utilisation. Des fichier `liste_suppr_ap` , `liste_suppr_av` peuvent également être utilisés.

Pour l'utiliser, il suffit de lancer le script `./vector.sh` ou d'ajouter les arguments *`-config vector/config.xml`* à la ligne de commande de DJIN ou de `Robot`.

Dans le fichier d'expressions régulières chaque enregistrement fait 3 lignes :

1. Nom du groupe de mots-clefs
2. Un caractère # suivi d'une tabulation et d'un chiffre qui est le type de groupe de mots-clefs (pour information seulement).
3. L'expression régulière (sans caractères + ou *)

[3] http://vizier.u-strasbg.fr/vizier/catmap.gif

Les expressions régulières ne doivent pas contenir de cardinalité infinie (+ ou *) mais seulement des fixes : **?** ou **{x,y}**.

Le résultat d'une recherche de mots-clefs n'est pas enregistré dans un fichier `objnames.csv` mais dans un fichier dont le chemin est obtenu en ajoutant `.csv` à celui de la liste de documents.

Ce fichier contiendra deux lignes pour chaque référence différente (bibcode, catalogue VizieR, ...).

- La première ligne contient la référence puis les numéros de groupes dont au moins un mot-clef a été détecté. Les champs sont séparés par des blancs
- La deuxième ligne contient le mot “OCCS:” suivi par les nombres d'occurrences pour chaque numéro de groupe de la première ligne, séparés par des blancs. Il y a donc autant de chiffres que dans la première ligne.

Exemple :

```
2008AstBu..63..357K 1 3 6 45 154 176 179 180 189 215 220 237 241 340
OCCS: 1 6 1 1 2 4 1 4 1 1 1 2 1 2
2009A&A...496..299P 1 3 6 45 154 176 179 180 189 215 220 237 241 340
OCCS: 1 6 1 1 2 4 1 4 1 0 1 2 3 2
2009MNRAS 3 148 179 199 202 226 242
OCCS: 5 0 1 1 2 0 1
2009MNRAS.394L..84G 3 154 179 180 221 285 353
OCCS: 12 2 1 10 1 1 14
```

Si deux références identiques se suivent dans la liste des documents, leurs nombres d'occurrences sont ajoutés.

Le fichier de configuration contient dans le paramètre d'extraction positionweights des coefficients par lesquels sont multipliés les nombre d'occurrences en fonction de leur type de position (titre, abstract, table, ...). Le résultat devant être un entier, cela explique qu'on puisse avoir la valeur 0 dans les chiffres de la deuxième ligne du fichier résultat. Les types de position non précisés ont par défaut le coefficient 1.

Table 1. Exemple des valeurs que peut contenir le paramètre positionweights :

Value	Result	Type de position
a	2.5	Abstract
c	0.5	Légende de figure ou table
d	0.5	Table
k	2.5	Mot-clef
s	2.0	Sous-titre
t	3.0	Titre

La liste des groupes de mots-clefs du fichier d'expressions régulières est enregistrée dans un fichier nommé `keys.txt` dans le même répertoire que la liste de documents. Il donne la correspondance entre les numéros de colonne du fichier précédent et les noms des groupes.

Pour les fichiers en local, les auteurs dont les noms ont pu être extraits sont enregistrés dans le fichier `cat_authors.txt` au format : `référence ; auteur 1, auteur 2, ...`. Ce fichier est utilisé par la recherche dans les catalogues de VizieR

Lorsque la liste des documents contient des noms de fichiers `ReadMe` (de catalogues VizieR) ceux-ci sont analysés de manière spécifique par la fonction `Annot.readTextFile` (appel à `Sortie.determinerTypeReadMe`) en tenant compte des règles d'écriture des fichiers ReadMe[4].

[4] http://vizier/doc/catstd-3.1.htx

2.2. Recherche de catalogues VizieR

L'idée est de rechercher dans la bibliographie des catalogues de VizieR les mots-clefs qui sont en général demandés par les utilisateurs de VizieR lorsqu'ils effectuent une recherche dans les catalogues.

2.2.1. Arborescence des catalogues en local

La liste des articles cités dans les catalogues est d'abord regroupée dans une arborescence en local dont tous les noms de fichiers sont dans une liste de validation (fichiers ReadMe et documents PDF identifiés par leur bibcode).

La liste complète des catalogues a été obtenue avec findcat -s .

Fichiers de l'arborescence :

- les fichiers ReadMe décrivant les catalogues (8199 fichiers)
- les fichiers PDF cités dans les références bibliographiques des catalogues (7870 fichiers nommés bibcode.pdf)
- un fichier de même nom et d'extension `.containstext` si du texte peut être extrait du document PDF (5652 fichiers)
- un fichier de même nom et d'extension `.txt` contenant le résultat du passage d'un OCR si le PDF ne contient que les images des pages (2394 fichiers)
- un fichier de même nom et d'extension `.html` lorsque l'extraction de texte a déjà extrait et exporté le contenu au format HTML.

Extrait de la liste des catalogues et documents :

```
...
I/203/;file:///data/vizier/cats/I/203/1994BICDS..44....9P.txt
I/203/;file:///data/vizier/cats/I/203/ReadMe
I/208/;file:///data/vizier/cats/I/208/1994A&AS..105..301R.txt
I/208/;file:///data/vizier/cats/I/208/ReadMe
I/294/;file:///data/vizier/cats/I/294/ReadMe
I/282/;file:///data/vizier/cats/I/282/2002A&A...395..347E.pdf
I/282/;file:///data/vizier/cats/I/282/ReadMe
...
```

Le programme DJIN est ensuite utilisé pour générer une liste de vecteurs correspondant aux occurrences des mots-clefs détectés dans ces documents. Les occurrences sont pondérées en fonction de la position dans le document (titre, abstract, mots-clefs, ...).

Lors de cette étape, DJIN génère les fichiers suivants :

- `list.txt.csv` : liste de vecteurs donnant l'occurrence de chaque groupe de mots-clefs dans chaque catalogue
- `keys.txt` : liste des mots-clefs issue du fichier liste
- `cat_authors.txt` : liste des auteurs de chaque catalogue issue des ReadMe

Extrait de la liste des vecteurs (les références sont des noms de catalogues et non plus des bibcodes) :

```
Num/Occurences
I/261/ 3 66 184 194 195 211 218
OCCS: 53 0 4 1 6 2 2
```

```
I/251/ 3 6 9 18 22 24 30 33 36 164 165 177 194 195 197 207 211 214 220 249 254 
OCCS: 3 2 6 1 29 7 2 8 2 1 7 15 1 3 1 3 1 0 1 1 12 105
...
```

2.2.2. Exploitation du résultat

Une fois ce fichier résultat rempli, le programme `FindCatalog` peut l'utiliser pour calculer le score de tous les catalogues pour une liste de mots donnée entrée dans la zone de saisie du moteur de recherche VizieR.

Voici la syntaxe :

`java -cp objectname.jar cds.vizier.FindCatalog -list` *`result_file.csv`* `-config` *`config.xml words_to_be_searched`*

L'option *`-config`* doit indiquer un fichier de configuration qui pointe (paramètre astrodir) vers le répertoire où se trouve le même fichier d'expressions régulières que celui qui a servi à obtenir le fichier résultat donné par l'option *`-list`*. Sinon les numéros de colonne du résultat ne correspondront pas à ce qui sera trouvé dans la chaîne de recherche.

Le programme effectue les actions suivantes :

- Les mots-clefs sont recherchés dans la chaîne de recherche et un vecteur en est déduit comme ceux du fichier de résultat.
- Un “produit scalaire” est effectué entre ce vecteur et ceux des catalogues, il donne un premier score.
- Les noms d'auteurs sont cherchés dans la chaîne de recherche. Chaque nom trouvé augmente le score du catalogue en fonction de son rang dans la liste des auteurs du catalogue.
- Les noms des catalogues eux-mêmes sont aussi cherchés et viennent augmenter les scores
- Si des coordonnées sont trouvées, l'utilisateur est orienté vers une recherche par coordonnées.
- La liste des vecteurs est affichée par score décroissant dans un tableau au format HTML.

L'argument *`-searchFile`* avec un nom de fichier peut être utilisé pour indiquer un fichier dans lequel chaque ligne est une chaîne de recherche. Le programme va effectuer toutes les recherches et en afficher la liste sur la sortie standard avec le nombre de catalogues trouvés. Cela permet de repérer les recherches ne fournissant aucun résultat et donc des mots à ajouter à la liste des mots-clefs.

Dans la fonction `FindCatalog.main`, le booléen `bDetail` indique d'ajouter une troisième colonne au tableau HTML qui permet d'afficher un lien vers une requête qui effectue le recalcul du détail pour chaque catalogue.

2.3. Recalcul du détail pour un catalogue donné

Comme les nombres d'occurrences des mots-clefs dans les documents d'un catalogue sont pondérés et additionnés, il n'est plus possible de connaître exactement le détail du calcul du score d'un catalogue pour une chaîne de recherche donnée.

Ce programme refait donc la recherche des mots-clefs dans les documents d'un catalogue pour afficher le détail de ce calcul dans un tableau HTML.

Syntaxe (les arguments sont obligatoires) :

`java -cp objectname.jar -list` *`vector/list.txt.csv`* `-config` *`vector/config.xml`* `-bib` *`J/AJ/107/2240/ bright star metallicity AJ/107 carney latham`*

L'argument -bib indique le nom du catalogue pour lequel le détail sera calculé.

Le reste de la ligne de commande représente la chaîne de recherche.

Pour cet exemple, le résultat est le suivant :

Table 2. Exemple du résultat du détail de calcul du score pour un catalogue donné

J/AJ/107/2240/					
Document	**Score**	**Keyword**	**Score**	**Position**	**Nb**
1994AJ....107.2240C.txt	96	Bright star	1	Abstract	1
		metallicity	95	Abstract	76
ReadMe	5	metallicity	5	Keyword	4
Author names	90	CARNEY	60	Author	1
		LATHAM	30	Author	2
Catalogue names	100	AJ/107			
Popularity	12			Popularity	551

Les nombres d'occurrences pour chaque type de position sont donnés pour tous les mots-clefs dans tous les documents du catalogue où ils apparaissent. Les noms d'auteur sont donnés avec leur rang dans la dernière colonne. Un nom de catalogue même partiel fait également augmenter le score. La popularité est prise en compte.

Les paramètres suivants interviennent dans le calcul du score. Les trois premiers sont en durs dans la classe `Catname` et les autres sont dans le paramètre positionweights du fichier de configuration.

Table 3. Paramètres utilisés pour calculer le score d'un catalogue

Paramètre	**Ex. de Valeur**	**Explication**
Author pertinence	60	Valeur ajoutée pour un nom d'auteur divisée par le rang de l'auteur
Name pertinence	100	Valeur ajoutée pour un nom de catalogue (contenant le caractère /)
Popularity pertinence	20	Le logarightme de la popularité + 1 est multiplié par ce nombre divisé par 10 et ajouté au score
Title weight	1.5	coefficient multiplicateur du nombre d'occurrences dans le titre
Keyword weight	1.25	coefficient multiplicateur du nombre d'occurrences dans les mots-clefs
Abstract weight	1.25	coefficient multiplicateur du nombre d'occurrences dans le résumé
Subtitle weight	1.0	coefficient multiplicateur du nombre d'occurrences dans les sous-titres
Text weight	0.5	coefficient multiplicateur du nombre d'occurrences dans le texte
Caption weight	0.25	coefficient multiplicateur du nombre d'occurrences dans les légendes de figure ou table
Table weight	0.1	coefficient multiplicateur du nombre d'occurrences dans les tables

2.4. Recherche des catalogues sur un site en local

Les fichiers suivants permettent de faire tourner la recherche des catalogues ainsi que le calcul du détail du score pour un catalogue donné en local. Ils se trouvent dans les sources à l'emplacement suivant : `~/objectname/djin/web/webapps/`

- `httpd.conf` : fichier de configuration du serveur Apache spécifique à ce sous-répertoire et permettant l'exécution de scripts en Perl.
- `search/index.html` : formulaire contenant la zone d'édition et le bouton permettant de lancer la recherche. Il contient également un lien vers une liste de recherches issue du log du moteur de VizieR et un lien vers la liste des mots-clefs recherchés dans les documents.
- `search/searchCompare.html` : liste de recherches issue du log du moteur de VizieR. Chaque lien contient un lien permettant de lancer la recherche en utilisant `FindCatalog` ou le moteur de VizieR.
- `search/cgi-bin/keywords.pl` : affichage de la liste des mots-clefs recherchés
- `search/cgi-bin/searchFrame.pl` : page contenant les deux cadres (un pour `FindCatalog` et un pour VizieR)
- `search/cgi-bin/search.pl` : lancement de la recherche par `FindCatalog` et affichage du résultat
- `search/cgi-bin/detail.pl` : lancement du calcul du détail du score pour un catalogue par `DetailCatalog` et affichage du résultat

3. Utilitaires

3.1. Génération de fichiers parfile de titres

Les fichiers parfile ayant servi à la génération des tableaux HTML contenant les titres et bibcodes des articles de SIMBAD (par le programme de lecture des fichiers parfile) ont eux-mêmes été générés par cet utilitaire qui récupère les informations depuis SIMBAD.

Les arguments du programme `TitleToFile` sont les suivants :

- *`-year`* permet de préciser les années de début et de fin (ex. : 2000..2001)
- *`-edit`* permet de se limiter à un journal
- *`-config`* permet d'indiquer un fichier de configuration
- *`-abstractDir`* donne un répertoire de destination pour le fichier parfile résultat

Au moins un des deux premiers arguments est obligatoire.

Le programme génère le script SIMBAD suivant :

```
format ref bib "%REFLIST(%B;%T\\n)"
query bib year=??? jnl=???
```

Où les “???” sont remplacés par les paramètres correspondants.

La réponse est ensuite analysée pour générer le fichier parfile contenant les champs “%R” et “%T” pour les bibcodes et les titres.

Traduction de noms d'objet entre les bases SIMBAD et NED

L'objectif est d'avoir un outil de traduction des noms d'objet de SIMBAD[1] vers les noms d'objet de NED[2] et inversement pour fournir aux utilisateurs de chaque base un lien depuis la description d'un objet vers la description du même objet dans l'autre base.

La stratégie retenue est d'utiliser les informations du Dictionnaire de Nomenclature[3] pour générer une table de correspondance entre les formats puis d'utiliser cette table pour effectuer les transformations en utilisant des remplacements par expressions régulières. La technique du remplacement permet de transférer dans un format de destination les informations chiffrées extraites de l'identificateur de départ.

1. Quelques chiffres sur les formats NED

A propos des informations sur les formats NED dans le Dictionnaire de Nomenclature (2008) :

- 2697 acronymes contiennent des informations sur le format dans NED (sur 15 264)
- 3802 formats NED sont présents (certains acronymes ont plusieurs formats)
- 224 formats NED correspondent exactement au format SIMBAD.
- 54 formats NED pour des acronymes SIMBAD en "-" (ce qui signifie que ces identificateurs ne sont pas dans SIMBAD)

Le fichier `ned-catal.txt` est l'équivalent du dictionnaire pour NED. En 2008, il contenait :

- 7000 lignes en 0> (format NED standard)
- 5500 lignes en 1> (alias)
- 4414 paragraphes de description d'acronymes
- 3365 formats NED avec un exemple correspondant nous ont été envoyés par NED par mail depuis 1998
- 3200 (95 %) de ces exemples existent réellement dans NED
- 17 seulement de ces exemples ne correspondent pas au format

> **Note**
> Il y a plus d'ambigüités dans les formats NED car il n'utilisent pas d'accolades pour préciser les lettres qui ne changent pas.

En 2008, le Dictionnaire de Nomenclature a été modifié pour inclure les informations de correspondance entre les formats NED et SIMBAD ainsi que les exemples d'identificateurs de NED correspondant aux formats (un par format).

Indication du format NED correspondant à un format SIMBAD :

[1] http://simbad.u-strasbg.fr/simbad/
[2] http://nedwww.ipac.caltech.edu/
[3] http://cds.u-strasbg.fr/cgi-bin/Dic-Simbad

```
%F {Ch} N=NGC 4151:[A77] Ch N
%F {G}N=NGC 4151:[A77] GN
%F {NEQ3}=NGC 4151:[A77] NEQ3
%F {NEQ3 z}N=NGC 4151:[A77] NEQ3 zN
```

Liste des formats de NED avec leur exemple pour un acronyme SIMBAD donné :

```
%B CDFN:[ABB2003] NNN,CDFN:[ABB2003] SNN,CDFS:[ABB2003] NNN,CDFS:[ABB2003] SNN
%b CDFN:[ABB2003] 501,CDFN:[ABB2003] S79,CDFS:[ABB2003] 326,CDFS:[ABB2003] S42
```

DJIN permet d'analyser et vérifier les informations NED du dictionnaire en utilisant l'option -dicoUrl

2. Compilation des programmes et génération de l'arborescence

L'arborescence placée dans le répertoire des source `~/objectname/idTranslator` contient les archives Java, la table de correspondance entre les formats, et les fichiers nécessaires aux fonctions de `simbad.jar` pour interpréter les identificateurs. Elle peut être générée en lançant la commande `ant package` depuis le répertoire des sources de `simbadned` : `~/objectname/simbadned`

Il est possible de lancer depuis le répertoire `~/objectname/idTranslator` des commandes en lignes pour la traduction des identificateurs dans les deux sens, pour les besoins d'une démonstration par exemple.

3. Génération de la table de correspondance entre les formats

Cette table contient sur chaque ligne un acronyme, un format, et une expression régulière correspondant aux identificateurs SIMBAD associés à un format NED.

La raison est que la génération des expressions du côté SIMBAD est un processus compliqué alors que l'on peut facilement créer les expressions pour les identificateurs NED à partir de leur format une fois que les parties fixes y ont été délimitées par des accolades. De plus, il avait été envisagé au départ de fournir la table de correspondance à NED pour qu'ils puissent générer eux-mêmes leurs identificateurs SIMBAD sans pour autant disposer de tout le Dictionnaire de Nomenclature.
Voici un extrait de la table de correspondance. Dans le fichier réel (`acroFormatNed`), les champs sont séparés par des “;” .

Table 1. Table de correspondance entre les formats SIMBAD et NED

Acronyme SIMBAD	Format SIMBAD	Expression régulière SIMBAD	Format NED
ACH	NN	A[cC][hH] ?(\d{1,3})	{ACH }NN
[BFT2007]	NNN	\[?[bB][fF][tT]2007 ?] ?(\d{1,4})	{NGC 2264:[FMS2006] }NNN
[BFT2007]	NNN	\[?[bB][fF][tT]2007 ?] ?(\d{1,4})	{NGC 6618:[BFT2007] }NNN
[FMS2006]	NNN	\[?[fF][mM][sS]2006 ?] ?(\d{1,4})	{NGC 2264:[FMS2006] }NNN
[ACM2000]	N-NN A	\[?[aA][cC][mM]2000 ?] ?(\d)[-](\d{1,3}) ?([A-Z]?)	[ACM2000] N-NNA

Le programme `DicoNed.` génère à partir du fichier main du dictionnaire la table contenant pour chaque acronyme SIMBAD une correspondance entre les formats SIMBAD et NED.

Il y a une ligne pour chaque format SIMBAD.

Si un acronyme contient plus de formats NED que de formats SIMBAD, alors des formats SIMBAD seront répétés car la table doit servir dans les deux sens.

L'expression régulière correspond à un identificateur SIMBAD. Elle contient des groupes entre parenthèses () qui permettront de placer les données chiffrées des formats SIMBAD aux bons endroits dans l'identificateur NED par une opération de remplacement.

C'est la fonction `DicoNed.acroIteratorFromMain` qui génère ces expressions régulières en appelant `Regular.genererRegExp` qui va parcourir le fichier `main` du dictionnaire comme pour les expressions régulières de DJIN.

Les exemples de NED sont également extraits du dictionnaire. Ils seront utilisés par `DicoNed.genererFormatsNed` pour déterminer les parties fixes des formats NED et les délimiter par des accolades.

L'instruction suivante à lancer dans le répertoire `~/objectname/idTranslator` permet de générer la table de correspondance `acroFormatNed` dans le répertoire de destination :

```
java -cp simbadned.jar:objectname.jar cds.data.ident.ned.DicoNed http://.../main Ned/greekSimbadNed ./Ned/
```

Il est aussi possible de lancer cette génération depuis le répertoire de DJIN :

```
java -cp simbadned.jar:objectname.jar cds.data.ident.ned.DicoNed http://.../main nomenclature/greekSimbadNed /tmp
```

4. Traduction des noms d'objet de SIMBAD vers NED

Ce programme utilise la table de correspondance pour traduire un identificateur SIMBAD en son équivalent dans NED.

La classe utilisée est : `SimbadNed` . La syntaxe est :

```
java -cp simbadned.jar:objectname.jar:simbad.jar -table . -coord "12 19 27.79 +05 02 50.4" "[SBM98] ACO 1516 J121927.79+050250.4"
```

Etant donné un identificateur et son acronyme, les étapes sont les suivantes :

- Chargement de la liste des acronymes et formats
- Recherche des formats SIMBAD et leur expression régulière pour l'acronyme
- Si une expression correspond, calcul de l'expression de remplacement à partir du format NED correspondant
- Application du remplacement pour donner l'identificateur NED
- Transformation des lettres grecques SIMBAD en leur équivalent pour NED
- Si le résultat n'est pas conforme au format de destination et que celui-ci contient des coordonnées, elles sont remplies avec les coordonnées de l'objet

SIMBAD affiche désormais un lien vers NED qui utilise le programme. Ce lien apparaît lorsque l'option Ned (External archive links) est cochée (elle l'est par défaut).

Exemple avec : M 31[4] (à la fin de la page dans "External archives")

5. Traduction des noms d'objet de NED vers SIMBAD

Ce programme utilise la table de correspondance pour traduire un identificateur NED en son équivalent dans SIMBAD.

La classe utilisée est : `NedSimbad` . La syntaxe est :

```
java -cp objectname.jar:simbadned.jar:simbad.jar
cds.data.ident.ned.NedSimbad -table . -coord "00 42 44.31 +41 16 09.4"
"MESSIER 31"
```

Etant donné un identificateur NED, les étapes sont les suivantes :

- Détermination de l'acronyme (début du nom)
- Chargement de la liste des acronymes et formats
- Recherche des formats NED pour l'acronyme
- Calcul des expressions régulières à partir des formats NED
- Si une expression correspond, calcul de l'expression de remplacement à partir de l'expression régulière SIMBAD correspondante
- Si aucune expression ne correspond, on peut recommencer les 3 étapes précédentes avec un acronyme plus court
- Application du remplacement pour donner l'identificateur SIMBAD
- Transformation des lettres grecques NED en leur équivalent pour SIMBAD
- Si le résultat n'est pas conforme au format de destination et que celui-ci contient des coordonnées, elles sont remplies avec les coordonnées de l'objet

SIMBAD peut désormais être interrogé avec les identificateurs NED. Voici comment construire l'URL de la requête :

- Le début est le même que pour les identificateurs SIMBAD : `http://simbad.u-strasbg.fr/simbad/sim-id?`
- Ajouter le paramètre `Ident=` suivi de tous les identificateurs NED de l'objet séparés par des “;”
- Ajouter le paramètre `&Coord=` (facultatif) suivi des coordonnées dans le système équatorial J2000.0. Elles doivent parfois être incluses dans l'identificateur.
- Ajouter le paramètre `&QueryType=ned` qui précise que ce sont des identificateurs NED à traduire

Les paramètres de l'URL doivent bien sûr être encodés.

[4] http://simbad.u-strasbg.fr/simbad/sim-id?Ident=M31#lab_external

Table 2. Exemples de requêtes à SIMBAD avec des identificateurs NED

Identificateur NED	Coordonnées	URL
ABELL 2063:[SPS89] 23[5]	15h22m54.3s +07d38m06s	http://simbad/simbad/sim-id?Ident=ABELL+2063%3A%5BSPS89%5D+23&Coord=15h22m54.3s%20%2B07d38m06s&QueryType=ned
KTS 76 ; [WPV85] 113[6]	3h53m22.5s -40d48m37s	http://simbad/simbad/sim-id?Ident=KTS%2076;%5BWPV85%5D%20113&Coord=3h53m22.5s%20-40d48m37s&QueryType=ned
[ABELL 2390:[NAM2006] P1 06_A[7]	21h53m35.3s +17d41m55s	http://simbad/simbad/sim-id?Ident=ABELL%202390%3A%5BNAM2006%5D%20P1%2006_A&Coord=21h53m35.3s%20%2B17d41m55s&QueryType=ned

Dans la dernière requête, les coordonnées sont utilisées pour créer l'identificateur traduit.

6. Tests d'intégration de la traduction des noms d'objet

Les tests `TranslationTest.testSimbadNed` et `TranslationTest.testIdSimbad` permettent de traduire de grandes listes d'identificateurs enregistrés dans les fichiers respectifs `~/objectname/idTranslator/Ned/testSimbadNed` et `~/objectname/idTranslator/Ned/testIdSimbad`.

Le premier fichier a été rempli en rassemblant tous les exemples des dictionnaires de SIMBAD et NED et le deuxième avec une sélection d'identificateurs demandés par les utilisateurs de SIMBAD. Ce sont des fichiers au format CSV séparés par des “;”. Les exemples NED et SIMBAD sont lus à chaque exécution pour pouvoir ajouter les nouveaux à la liste.

Les champs sont les suivants :

1. Identificateur SIMBAD
2. Identificateur NED
3. Exemple SIMBAD
4. Acronyme SIMBAD
5. Exemple NED
6. Coordonnées
7. Test de la transformation de SIMBAD vers NED (correct = "1")
8. Test de la transformation de NED vers SIMBAD (correct = "1")
9. L'exemple NED a-t-il déjà été vérifié dans NED ? (oui = "1")
10. L'exemple SIMBAD a-t-il déjà été vérifié dans SIMBAD ? (oui = "1")

Chaque ligne représente un élément de test stocké dans un objet `TestElement` . Selon la terminologie choisie, les identificateurs ont été vérifiés et existent dans leur base, tandis que les exemples sont issus d'un dictionnaire ou de la traduction d'un identificateur de l'autre base.

Les traitements effectués sont les suivants :

- Vérification de l'exemple SIMBAD (en utilisant l'acronyme, les coordonnées sont récupérées)
- Traduction de l'identificateur SIMBAD en exemple NED (en utilisant éventuellement les coordonnées)
- Vérification de l'exemple NED (ses coordonnées sont récupérées)
- Traduction de l'identificateur NED en exemple SIMBAD
- Contrôle que la traduction de l'identificateur SIMBAD donne bien l'équivalent NED
- Contrôle que la traduction de l'identificateur NED donne bien l'équivalent SIMBAD
- Un tableau HTML est généré où les cases vides ou fausses (pour les booléens) sont affichées en rouge
- Calcul des totaux et pourcentages à la fin du tableau

A moins qu'une exception soit levée, ces tests réussissent toujours, il faut consulter le résultat sur la sortie standard ou le fichier HTML généré dans le même répertoire que le fichier `acroFormatNed` (l'emplacement de ce fichier est stocké à sa génération dans la préférence “tableDir” de la classe `DicoNed`.

Les exemples du fichier `testIdSimbad` sont sencés tous fonctionner.

Le test `ReentrantTest.testIdSimbad` reprend ce dernier fichier et en contrôle les traductions dans le sens SIMBAD vers NED en parallèle dans 30 fils d'exécution (threads) simlultanés pour tester l'éxécution en mode multi-tâche. Le test échoue dès qu'une traduction ne correspond pas au résultat désiré.

Suites à donner

1. Développements à poursuivre

La table de correspondance pour la traduction des noms d'objet entre les base SIMBAD et NED devrait idéalement être générée à chaque mise en ligne d'une nouvelle version du Dictionnaire de Nomenclature. Cette génération ainsi que l'exécution du test `ReentrantTest.testIdSimbad` devraient donc être ajoutées à la liste des actions automatiquement effectuées à cette occasion. Au 08/02/2010, le test relevaient un certain nombre d'erreurs qui proviennent du dictionnaire.

L'arborescence des catalogues de VizieR devrait pouvoir être complétée à chaque nouvelle mise à disposition de catalogue. Le script `~/objectname/djin/script/updateVizierTree.pl` est développé dans ce but. Les documents PDF qu'elle contient et qui ne contiennent que des images devraient être analysés avec un OCR qui gère les textes sur deux colonnes, ou alors les images générées par le script `~/objectname/ocrVizierPdf.pl` devraient être découpées pour ne contenir qu'une colonne.

La liste des mots-clefs recherchés par le programme qui génère les vecteurs de nombres d'occurrences peut être complétée, notamment avec les noms des différents téléscopes et instruments existants. Elle contient déjà les noms des instruments et téléscopes spatiaux.

Un script simple permettant de télécharger tous les articles d'un volume de MNRAS peut être développé. Exécuté sous Linux, il permettrait ensuite de fournir ces articles à Philippe Vonflie dont le poste tourne sous Windows et ne peut pas récupérer directement les documents PDF à partir des bibcodes.

Liste de bugs :

- Changement des couleurs de liens si vérification manuelle des noms d'objet : lorsque la préférence Automatically check names in SIMBAD n'est pas cochée, les noms d'objet surlignés dans le texte extrait restent en orange même après que la vérification ait été lancée manuellement.
- Griser le bouton inutile si paramètre d'extraction manquant pour localiser l'URL des documents PDF ou HTML
- Supprimer les paramètres d'extraction PDF ou HTML qui ne fonctionnent pas dans le fichier de configuration
- Gérer le cas de l'objet Taurus

2. Promotion et publication

DJIN gagnerait à être connu dans d'autres laboratoires. Sa promotion pourrait se faire dans le cadre du projet Plume[1].

Publication des sources : le logiciel DJIN et la traduction des identificateurs ayant été développés dans le cadre du programme européen VO-Tech, leur code source devrait être publié sous licence open source et hébergés sur un site tel que SourceForge[2] ou Google Code[3].

[1] http://www.projet-plume.org/
[2] www.sourceforge.net
[3] www.code.google.com

3. Fichiers à conserver

3.1. Articles de 2008

Une liste de 7443 bibcodes se trouve dans le fichier `uparfile-2008` qui contient également les titres, abstracts, et mots-clefs des articles. Ce fichier a été utilisé pour comparer les détections faites par le programme de détection dans les fichiers parfile avec les informations entrées par les documentalistes sur les mêmes articles. Le résultat de cette comparaison est dans ce tableur sur le site interne[4] qui contient une ligne par bibcode contenant des noms d'objet dans le titre, le résumé ou les mots-clefs.

Le même travail de comparaison a été effectué sur les articles complets dont les fichiers PDF ont donc été récupérés et placés dans une arborescence contenant des sous-répertoires pour les journaux A&A, AJ, ApJ, IBVS, MNRAS, PASJ, PASP plus `misc` pour les petits journaux. Le répertoire `script` contient les scripts de mise à jour générés. Le fichier `bibcodes_file.txt` contient la liste de tous les bibcodes avec le chemin du document PDF correspondant. Le résultat de la comparaison est dans cet autre tableur[5] qui contient une ligne par référence contenant des objets.

Les articles de 2OO8 ont également tous été annotés en utilisant le programme de recherche des noms déjà entrés et placés dans une arborescence `2008annotated` équivalente.

3.2. Bibliographie de VizieR

L'arborescence des catalogues de VizieR est utilisée par la recherche de catalogues et contient les articles en PDF de la biliographie des catalogues mais également les résultats de l'extraction de texte ou du passage d'un OCR (anciens fichiers) ainsi que la description (ReadMe) des catalogues.

3.3. Documentation

Voici la liste des documentations disponibles pour les différents programmes réalisés :

- Fichiers du répertoire `~/objectname/djin/docbook/`
  - Ce fichier : la source est : `djinDoc.xml`, elle permet de générer les versions PDF et HTML
  - Documentation utilisateur : la première source était `ObjectNameRecognitionFR.htx` et était transformée en `doc.htm` par le programme `cgiprint`. Elle a maintenant été convertie au format Docbook par le script `htxToDocbook.pl` en : `ObjectNameRecognitionFR.xml`.
  - Documentation utilisateur anglaise : la première source était `ObjectNameRecognitionEN.htx` et était transformée en `docEN.htm` par le programme `cgiprint`. Elle a maintenant été convertie au format Docbook en : `ObjectNameRecognitionEN.xml`.
- Il y a un Twiki consacré à DJIN[6]
- Et un autre au lien SIMBAD - NED[7]
- Les fichiers volumineux ou nombreux ne pouvant pas être intégrés au Twiki sont sur ce site interne[8] (documentation des sources, résultats de détections, diagrammes, ...)

[4] http://snob.u-strasbg.fr/~bonnin/compTitAbs.ods
[5] http://snob.u-strasbg.fr/~bonnin/comp2008pdf.ods

Les sources des différents diagramme de ce document sont dans `~/objectname/djin/docbook/diagramme/` et ont été réalisés avec BOUML[9]. Certains diagrammes plus anciens sont faits avec UMLet[10] ou dia[11].

Les spécifications techniques des différentes versions du format PDF sont disponibles chez Adobe : " PDF Reference[12] ". Plus la version est récente et plus elles sont volumineuses...

[9] http://sourceforge.net/projects/bouml/
[10] http://www.umlet.com/
[11] http://www.gnome.org/projects/dia/
[12] http://www.adobe.com/devnet/pdf/pdf_reference_archive.html

Réflexion sur les choix et difficultés

1. Extraction de texte depuis les documents PDF

Dès le départ, les documents PDF ont été préférés aux documents HTML car le format PDF est beaucoup plus pérenne. Les éditeurs ne se jettent pas sur les dernières fonctionnalités introduites dans le format PDF (multimédia, pièces jointes, données 3D) et utilisent en général d'anciennes versions de ce format.

Certaines publications (surtout les anciennes) ne sont disponibles qu'en PDF.

Le contenu des articles en HTML est souvent réparti dans plusieurs fichiers (figures, tableau, ...).

Les documents en HTML ne peuvent en général pas s'imprimer directement.

La structure des documents en HTML fait souvent appel à une arborescence très compliquée qui nécessite l'utilisation de transformations XSLT pour en extraire le texte.

La librairie PDFBox[1] a été choisie car :

- Elle permet de lire les documents PDF.
- Elle permet également de les modifier et d'en créer.
- Elle est écrite en Java.
- Elle est libre (licence open source Apache).
- Elle offre une couche d'abstraction supplémentaire permettant d'utiliser les fonctionnalités du format PDF sans devoir manipuler les éléments de base des fichiers PDF (annotations, marque-pages, fonctions de dessin, ...)

Mais elle ne permet pas de récupérer le dessin d'un caractère décrit graphiquement (polices incorporées). C'est pourquoi la bibliothèque JPedal[2] a aussi été utilisée. Mais ce noyau open-source d'un produit commercial ne peut manipuler que les éléments de base de fichiers PDF.

Certains éditeurs (comme ApJ) écrivent tout l'article en utilisant les polices incorporées. Dans ce cas la description des caractères doit être utilisée pour extraire ceux qui ne sont pas reconnus graphiquement car reconnaître tout l'alphabet graphiquement (et pas seulement les symboles et lettres grecques) rendrait l'extraction beaucoup trop lente. C'est ce que fait le paramètre showothersymbol.

Les paramètres d'extraction permettant de régler l'extraction de texte sont stockés dans le fichier de configuration

Il y a une certaine hiérarchie entre les configurations puisque les paramètres codes, symbols, positionweights de la configuration générique sont aussi valables dans les configurations spécifiques à un journal sauf pour les valeurs qui seraient dans une configuration spécifique. Pour le paramètre astrodir, seule la valeur de la configuration générique est prise en compte.

Certains paramètres ne servent pas à l'extraction de texte mais sont quand même dans ce fichier soit pour avoir une valeur spécifique au journal comme bibcodestring, soit parce qu'ils ne sont pas destinés à être modifiés par les utilisateurs, contrairement aux préférences. Leurs valeurs seront donc remplacées par celles de l'installation de référence à chaque mise à jour.

[1] http://www.pdfbox.org
[2] http://www.jpedal.org

2. Impression des annotations PDF

Par défaut, les annotations ajoutées dans un document PDF apparaissent bien à l'écran mais pas sur la version imprimée.

Le format PDF prévoit un indicateur pour préciser qu'une annotation est imprimable. Celui-ci est bien positionné par `Annot.marquerPosition` mais ne semble pas fonctionner. C'est pourquoi la fonction Add printable annotations [15] a été créée.

3. Extraction du texte des documents HTML

Les éditeurs de certains journaux comme NewA et IBVS ne fournissent pas de version PDF exploitable de leurs articles car les mots sont mélangés et l'information permettant de les avoir dans l'ordre n'est pas présente.

La solution serait alors de classer les groupes de caractères de chaque page par position mais il resterait à résoudre le problème des présentations sur plusieurs colonnes.

Une autre possibilité pour rendre les PDF inexploitables serait de n'utiliser que des polices incorporées et de ne pas fournir la vraie description de chaque caractère, ce qui obligerait à reconnaître graphiquement tous les caractères utilisés (ce que fait un être humain).

C'est pourquoi le texte est alors extrait des documents HTML malgré leur structure compliquée.

Les feuilles de style XSL (dont le nom est dans le paramètre htmlfilter) sont utilisées car elles permettent de garder configurable cette extraction sans la programmer en dur.

4. Navigation sur Internet

4.1. Avec DJIN

DJIN permet une navigation limitée sur Internet puisque lors de la recherche du contenu d'un article en PDF, si une page HTML est rencontrée, elle est affichée et l'utilisateur peut cliquer sur les liens jusqu'à trouver le document PDF.

Pour trouver l'URL d'un article en PDF, DJIN utilise en général le serveur bibliographique ADS[3].

La raison est que ce service permet d'accéder au contenu PDF en ajoutant simplement le bibcode comme paramètre d'une URL fixe obtenue par le paramètre urlbibcode.

Exemple : `http://cdsads.u-strasbg.fr/cgi-bin/nph-data_query?link_type=ARTICLE&bibcode=2010AJ....139..216L`

Le même exemple avec le serveur biliographique du CDS donne :

`http://cds.u-strasbg.fr/cgi-bin/nph-doi?2010AJ....139..216D`

qui oblige l'utilisateur trouver le lien vers le contenu PDF dans la page obtenue.

La navigation sur Internet avec DJIN trouve parfois ces limites sur les sites de certains éditeurs. Par exemple lorsque l'accès au contenu n'est possible qu'après exécution de code Javascript.

Par exemple, sur le site de Interscience qui héberge les MNRAS, l'URL suivante permet d'accéder au contenu PDF avec un navigateur :

`http://adsabs.harvard.edu/cgi-bin/nph-data_query?link_type=ARTICLE&db_key=AST&link_type=ARTICLE&bibcode=2009MNRAS.400L..99T`

[3] http://cdsads.u-strasbg.fr/

Alors qu'avec DJIN une page d'erreur est affichée.

Une manière de contourner ce problème est d'utiliser un utilitaire comme `wget`. Une ligne de commande issue du paramètre urlcommand est complétée avec le bibcode désiré et un fichier de destination puis exécutée. Le résultat contient alors le contenu PDF de l'article. Malheureusement ce système ne fonctionne pas sous Windows, un utilitaire équivalent (“wget.exe”) n'ayant pas donné satisfaction.

4.2. Avec Firefox

Lorsque le contenu de l'article à récupérer est sous forme HTML et que DJIN ne peut pas le récupérer, une autre solution peut consister à modifier le navigateur pour qu'il puisse envoyer l'article à traiter directement à DJIN.

C'est le but de l'extension SAMP pour Firefox.

Malheureusement, celle-ci ne fonctionne pas sur tous les postes car les modules inclus dans Firefox peuvent être très différents d'une installation à l'autre et très peu d'information est disponible sur ceux-ci. En particulier le module XML-RPC[4] utilisé par l'extension SAMPDocument[5].

Sébastien Derriere a rencontré les mêmes problèmes en développant l'extension PLASTIC[6].

5. Sauvegarde de l'état courant

Actuellement, DJIN ne peut pas être mis à jour si un état courant a été sauvegardé sous peine de ne plus pouvoir le restaurer. De plus, un seul état courant peut être sauvegardé en même temps.

Cette sauvegarde utilise la sérialisation des objets de Java qui permet d'enregistrer facilement les informations contenues dans un objet ainsi que les objets membre qu'il contient. Mais si la définition des objets a changé et que de nouveaux membres ont été ajoutés, la relecture des informations échouera. Ce problème se serait aussi posé (mais moins souvent) si les informations avaient été enregistrées dans un autre format car des versions de format auraient forcément dues être gérées pour tenir compte de nouvelles fonctionnalités.

Le fait de n'autoriser qu'un seul état courant permet d'éviter qu'un trop grand nombre de sauvegardes puisse empêcher longtemps d'effectuer une mise à jour si le besoin s'en fait sentir.

Si cette fonctionnalité devait être demandée, il serait assez facile de permettre plusieurs sauvegardes en demandant un nom de sauvegarde au moment de l'enregistrement et en proposant les différentes possibilités s'il y en a plusieurs à la restauration. Les scripts qui vérifient la présence de l'état courant (`onrinstall.sh`) devront être modifiés en conséquence.

[4] http://www-archive.mozilla.org/projects/xmlrpc/
[5] https://addons.mozilla.org/fr/firefox/addon/9732
[6] https://addons.mozilla.org/fr/firefox/addon/6783

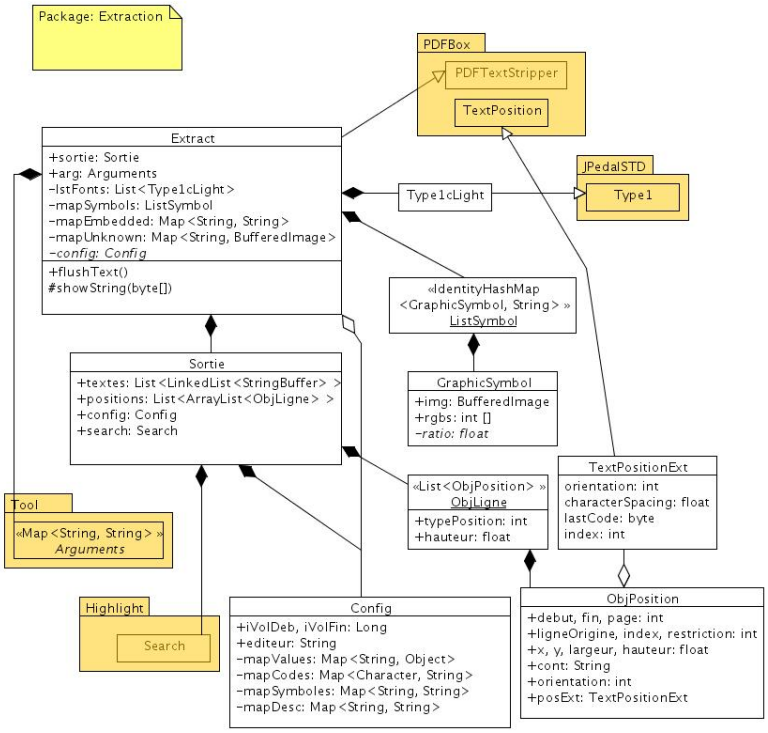
Liste des classes

1. Package cds.ObjectName.Extraction

Table 1. Package cds.ObjectName.Extraction

Config	Objet contenant les informations de configuration, en particulier les paramètres d'extraction de texte
Extract	Classe principale de l'extraction de texte à partir d'un document PDF ou texte ou HTML. Surcharge les fonctions flushText (récupération du texte dans l'ordre) et showString (orientation et position des caractères) de PDFBox
ExtractPos	Extraction de toutes les positions (groupes de caractèrs) d'une page d'un document PDF
GraphicSymbol	Image graphique d'un symbole en police incorporée. La fonction d'égalité permet de comparer graphiquement avec un autre symbole
HtmlCallback	Extraction du texte d'un document HTML. Surcharge les fonctions de javax.swing.text.html.HTMLEditorKit.ParserCallback pour parser le contenu d'un document HTML.
HtmlToUnicode	Conversion des caractères spéciaux HTML en leur équivalent Unicode. La table de correspondance est dans le fichier `unicode.html` . Permet de calculer la position des balises de mise en valeur des noms d'objet dans les document HTML
ListSymbol	Table de correspondance entre les images graphiques des symboles et leurs signification
ObjLigne	Liste contenant les positions (groupes de caractères des documents PDF) d'une ligne de texte extrait
ObjPosition	Informations sur la position d'un caractère ou groupe de caractères dans le texte extrait
Sortie	Texte extrait d'un document PDF ou HTML. Contient séparément dans deux tableaux les lignes de texte et les informations sur les positions (=groupes de caractères). Chaque tableau contient une liste pour le texte normal et une pour le texte des notes, légendes de figures et tableaux. Traitement du texte extrait.
TextPositionExt	Dérivée de TextPosition (position dans le document PDF au format PDF) avec en plus la notion d'orientation et d'espacement entre caractères
Type1cLight	Equivalent de la classe Type1C de la librairie JPedalSTD (http://www.jpedal.org/) dans lequel la fonction readType1CFontFile est publique pour pouvoir être appelée par l'extraction de texte
XHtmlCallback	Transformation d'un document HTML en XHTML. C'est un parser HTML qui réécrit le document HTML en respectant la norme XHTML

Figure 1. Diagramme des classes du paquet Extraction

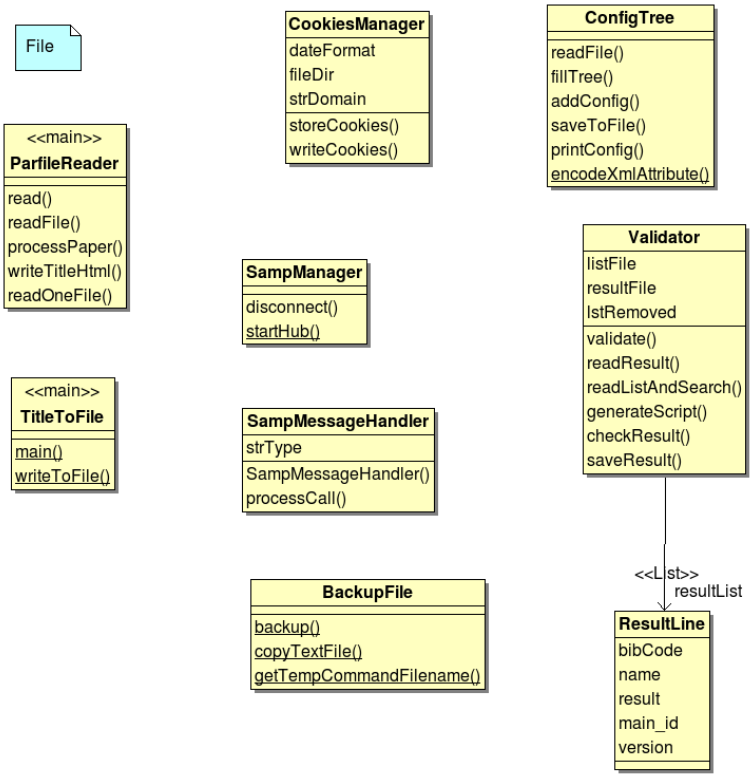


2. Package cds.ObjectName.File

Table 2. Package cds.ObjectName.File

BackupFile	Classe effectuant une sauvegarde tournante de fichiers texte contenant les scripts de mise à jour
ConfigTree	Arborescence hiérarchisée des paramètres de configuration, lecture / écriture dans un fichier XML
Cookie	Classe interne représentant un cookie
CookiesManager	Gestion de fichiers de cookies enregistrés sur le disque pour intégrer un comportement de navigateur web au programme
NotesToHtml	Detection of non tagged object names in SIMBAD notes (Writes an HTML table to standard output)
ParfileReader	Classe de lecture des fichiers parfile d'un répertoire contenant les bibcode, résumé, titre, et mots-clefs et de génération de fichiers de commande pour entrer les noms d'objet trouvés. Peut écrire les titres dans des tableaux au format HTML en fonction des propriétés des noms d'objet détectés.
ResultLine	Ligne de résultat du fichier d'exportation des résultats lors du traitement automatique d'une liste de documents. Il y a une ligne pour chaque nom d'objet dans un article.
SampManager	Gestion des messages SAMP. Lancement des traitements pour les messages SAMP reçus.
SampMessageHandler	Classe de traitement des messages SAMP reçus (classe privée du fichier `SampManager.java`)
TitleToFile	Envoi d'une requête à SIMBAD pour récupérer des titres d'articles et générer un fichier parfile contenant les bibcodes (%R) et les titres (%T). Ce fichier pourra être lu par le programme ParfileReader
XmlProcessor	Opération des transformations par XSLT sur les fichiers XML et XHTML

Figure 2. Diagramme des classes du paquet File

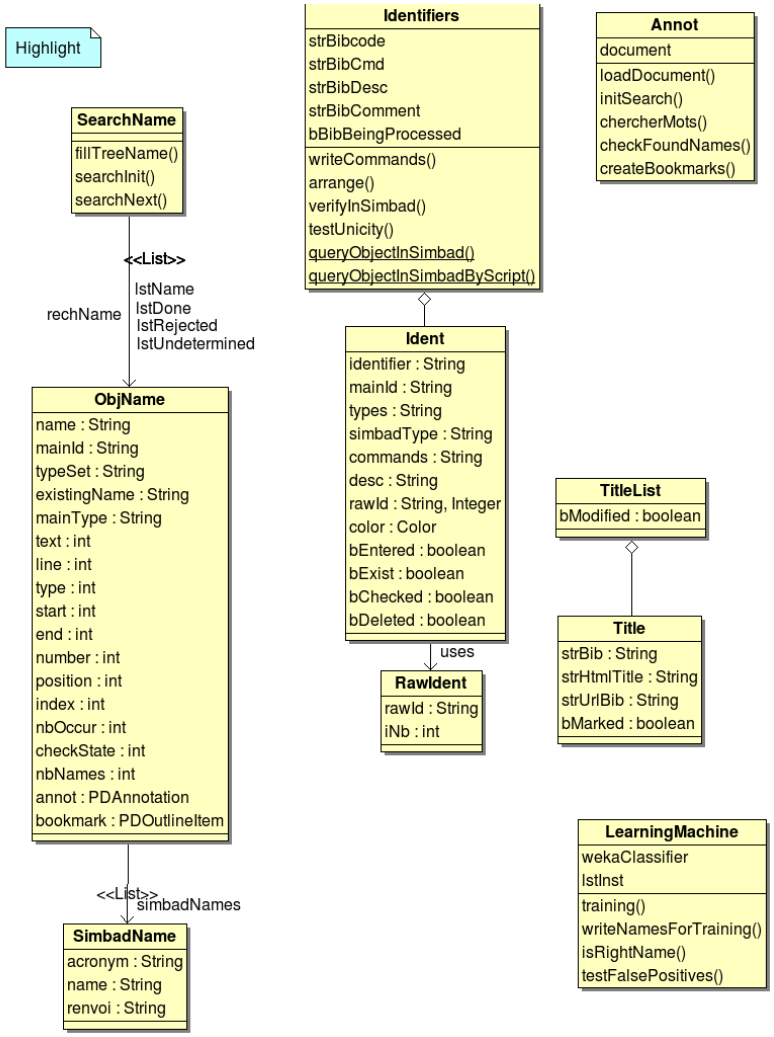


3. Package cds.ObjectName.Highlight

Table 3. Package cds.ObjectName.Highlight

AcroFreq	Lecture des raw id et identificateurs correspondants pour déterminer la fréquence des différents acronymes pour l'acronyme d'un nom donné. Génération du fichier acro_freq à partir du résultat d'une requête SIMBAD du genre : select ref_raw_id, ref_norm_id from has_bib_ref h , bib_ref b where oidbib=oidbibref and bibcode>'2009' and ref_raw_id is not null and ref_norm_id is not null. Ce programme a été utilisé pour générer le fichier acro_freq
Annot	Classe d'ajout des annotations, fenêtres pop-up et marque-pages.
Ident	Identificateur avec une couleur d'affichage et une liste de raw id avec leur nombre d'occurences. Eléments de l'objet liste Identifiers
Identifiers	Liste d'identificateurs devant être ajoutés à une référence bibliographique dans SIMBAD. Contient des objets Ident. Contient également les informations sur le bibcode.
LearningMachine	Objet de gestion du mécanisme d'apprentissage. Ecrit le fichier d'apprentissage au format ARFF. Parcours la liste des noms d'objet détectés pour vérifier ceux à rejeter.
ObjNameCompare	Comparateur de noms d'objet trouvés en fonction uniquement de leur position. Ce qui permet de les trier en vue de l'exportation du texte extrait avec les noms taggés
ObjNameCompExisting	Comparateur permettant de classer les noms d'objet à partir de leur nom SIMBAD. Utilisé par la recherche des noms d'objets à partir des raw id entrés dans SIMBAD.
ObjNameCompIndex	Comparateur permettant de classer les noms d'objet à partir de leur position dans le fichier (HTML) de départ en commençant par la fin. Utilisé lors de l'annotation de documents HTML
ObjName	Nom d'objet trouvé dans un texte avec les informations sur sa position, son état après vérification dans SIMBAD, les expressions régulières ayant permis sa détection, les noms SIMBAD auxquels il peut correspondre d'après le dictionnaire, les annotations et marque-page PDF correspondant.
PDFList	Liste de noms d'objet sous forme de document PDF pour pouvoir imprimer la liste des noms d'objets détectés dans un article
RawIdent	Nom d'objet tel que trouvé dans les documents avec son nombre d'occurrences. La fonction d'égalité est indépendante de la casse.
SearchEntered	Recherche dans le document des noms d'objets tels qu'ils sont déjà entrés dans SIMBAD en tenant compte des raw id, positions, nombre d'occurrence
SearchName	Recherche de noms d'objets dans le texte extrait
SimbadName	Nom d'objet dans Simbad avec leur acronyme et leur renvoi issus du dictionnaire. Le score permet de les classer dans la liste des identificateurs proposés au moment de l'ajout dans la liste des identificateurs.
TitleList	Liste de titres de références bibliographique avec les noms d'objet mis en valeur (programmes YesNo et Simref)
Title	Titre d'une référence bibliographique dans laquelle les noms d'objet sont mis en valeur. Elément des liste présentées par le programme YesNo

Figure 3. Diagramme des classes du paquet Highlight



4. Package cds.ObjectName.Ihm

Boîtes de dialogues et autres classes assurant le fonctionnement de l'interface homme-machine.

ActionAddAllNames	Action d'ajouter tous les noms d'objet de l'arborescence dans la liste des identificateurs.
ActionAddFromClipboard	Action d'ajout d'une liste de noms d'objet provenant du presse-papier directement dans la liste d'identificateurs
ActionAddName	Action d'ajout d'un nom d'objet dans la liste des identificateurs. Une liste d'identificateurs possibles est éventuellement proposée à l'utilisateur
ActionAddRegularExpression	Ajout d'une expression régulière dans l'arbre de recherche et relancement de la recherche de noms d'objet
ActionAddSelectedNames	Action d'ajouter les noms d'objet sélectionnés de l'arborescence dans la liste des identificateurs. Une liste des dentificateurs possibles est proposée à l'utilisateur s'il y en a plusieurs
ActionAPropos	Message A Propos
ActionBackToDocument	Action de retour vers le document extrait après avoir affiché une description d'objet ou de bibcode ou un résultat de traitement du dictionnaire, ...

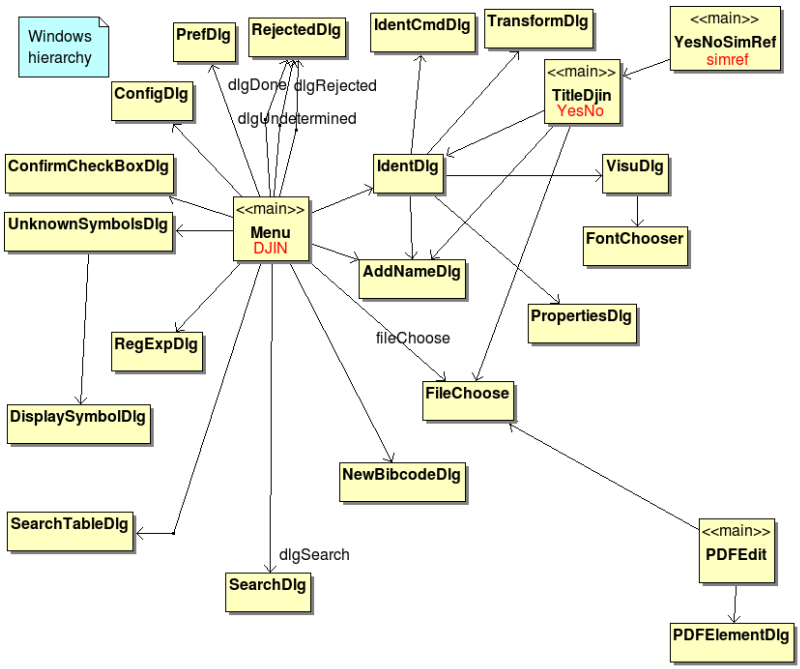
ActionBibcodeSimbad	Action d'interrogation de SIMBAD pour un bibcode. La description est affichée dans la fenêtre principale ou dans un navigateur
ActionCheckNames	Action de vérification dans SIMBAD des noms trouvés
ActionClearCurrentState	Action d'effacer le précédent état courant du programme sauvegardé
ActionClearNames	Action de remise à zéro de la liste des identificateurs
ActionConfig	Action d'ouverture de la boîte de dialogue de configuration des paramètres d'extraction
ActionCreateBibcode	Action de création d'un nouveau bibcode depuis le texte du presse-papier. Le texte est analysé en fonction d'un format donné par le paramètre bibcodestring. Après validation d'une boîte de dialogue, des commandes de mise à jour sont ajoutées au bibcode associé à la liste des identificateurs.
ActionDeleteName	Action de suppression d'un nom d'objet de l'arborescence. Le texte extrait affiché est réactualisé ainsi que les annotations et marque-pages du document PDF
ActionDeleteNamesAccordingARFF	Action de suppression des noms d'ojbet en fonction d'un fichier ARFF. Lorsque l'option -training est présente, cela permet de reporter les informations concernant les noms d'objet d'un bibcode depuis un fichier de données d'apprentissage (.arff) vers un autre.
ActionDisplayDoneNames	Action d'affichage de la liste des noms d'objet déjà traités dans une fenêtre enfant
ActionDisplayRejectedNames	Action d'affichage de la liste des noms d'objet rejetés dans une fenêtre enfant
ActionDisplayUndeterminedNames	Action d'affichage de la liste des noms d'objet indéterminés dans une fenêtre enfant
ActionEditCopy	Action de copier le texte sélectionné de la zone de texte dans le presse-papier (menu contextuel)
ActionEditSelectAll	Action de sélectionner tout le contenu de la zone de texte (menu contextuel)
ActionExportText	Action d'exportation du texte dans un fichier ASCII au format texte ou HTML
ActionFormats	Action de création de la liste d'expressions régulières à partir des formats du fichier main, du fichier des amas, de la liste des NAME(...) et p.(...) de SIMBAD
ActionHelp	Action d'affichage de l'aide dans un navigateur
ActionLargerSearch	Action de lancement d'une recherche supplémentaire avec un préfixe
ActionManageIdentifiers	Action d'ouverture de la boîte de dialogue des identificateurs (après vérification du bibcode)
ActionMarkNameAsDone	Action de marquage d'un nom d'objet de l'arborescence comme étant traité (ou bien indéterminé)
ActionMenu	Classe abstraite contenant une référence à l'objet Menu et dont dérivent toutes les classes d'action
ActionOpen	Action d'ouvrir un fichier PDF et d'en extraire le texte, de rechercher les noms d'objet, de vérifier éventuellement les noms dans SIMBAD

ActionOpenBibcode	Action d'ouverture d'un document PDF ou HTML à partir d'un Bibcode avec extraction et recherche
ActionOpenList	Action d'ouverture d'une liste de BibCode. Le résultat est affiché à raison d'une ligne par bibcode dans la fenêtre principale
ActionOpenNextBibcode	Action d'ouverture du bibcode suivant. Le nombre de page du document actuel est utilisé pour calculer ce bibcode.
ActionOpenText	Action d'ouverture d'un texte depuis le presse-papier ou un fichier ou une URL, avec recherche des noms d'objet
ActionOpenUrl	Action d'ouverture d'un document PDF à partir d'une URL avec extraction et recherche
ActionQuit	Action de quitter
ActionRemoveRegularExpression	Suppression d'une expression régulière de l'arbre de recherche et relancement de la recherche des noms d'objet
ActionRequestSimbad	Action d'interrogation de SIMBAD pour un nom d'objet. La description est affichée dans la fenêtre principale ou dans un navigateur
ActionSaveCurrentState	Action de sauvegarder l'état courant du programme
ActionSavePdf	Action d'enregistrement du document PDF ou HTML annoté
ActionSearchEntered	Action de recherche dans SIMBAD des noms d'objet présents sur la référence avec leurs raw id et mise en valeur dans le document.
ActionSearchInExtractedText	Action de lancement d'une recherche dans le texte extrait
ActionSearchInTable	Action de lancement d'une recherche supplémentaire avec un préfixe (factultatif) dans une colonne d'une table
ActionSearchWithPrefix	Action de lancement d'une recherche supplémentaire avec un préfixe
ActionUnknownSymbols	Action d'affichage de tous les symboles graphiques inconnus (non reconnus lors de l'extraction) du document
ActionUpgrade	Action de mise à jour automatique de DJIN
ActionViewNameList	Action de visualisation de la liste des noms d'objets trouvés sous forme d'un fichier PDF
ActionViewPdf	Action de visualisation du document PDF (ou HTML) annoté dans Acrobat Reader (ou un navigateur)
AddNameDlg	Fenêtre d'ajout d'un nom d'objet dans la liste des identificateurs. Les différents noms d'objet possible d'après le dictionnaire sont proposés
BibcodeListener	Classe de formattage du Bibcode (=code bibliographique) saisi dans la zone d'édition et autres traitements sur les bibcodes. Calcul de l'URL du contenu d'un article
ComboBoxRenderer	Affichage d'une ligne de liste déroulante dont la couleur dépend de l'existence de l'objet dans Simbad et qui contient le type d'objet donné par Simbad
DetailSelectionListener	Evènement de sélection d'une ligne de détail d'une table de codes
ConfigDlg	Fenêtre de configuration des paramètres de l'extraction de texte (et autres)
ValueSelectionListener	Evènement de sélection d'une valeur dans la liste des paramètres
ConfirmCheckBoxDlg	Boîte de dialogue affichant un message de confirmation et une case à cocher pour ne plus poser la question

DisplaySymbolDlg	Boîte de dialogue permettant de présenter à l'utilisateur un symbole graphique pour lui en demander la signification. L'image du symbole est alors enregistrée et sera utilisée pour comparer les symboles inconnus lors de l'extraction de texte
FileChoose	Boîte d'enregistrement de fichier affichant un message en cas de remplacement de fichier existant
FontChooser	A font selection dialog. Note: can take a long time to start up on systems with (literally) hundreds of fonts.
IdenListTransferHandler	Classe permettant le transfert par glisser-déplacer de listes d'identificateurs
IdentListTransferable	Classe contenant une liste d'identificateurs à regrouper et implémentant l'interface java.awt.datatransfer.Transferable
IdentCmdDlg	Edition des commandes de mise à jour pour un identificateur ou le bibcode. La partie haute contient la description issue de SIMBAD et la partie permet d'éditer les commandes
IdentDlg	Fenêtre de gestion de la liste des identificateurs
JTextFieldLimitedDoc	Extension du Document de Swing permettant d'avoir des JTextField dont la longueur du texte est limitée. On peut également n'y autoriser que des chiffres. Utilisation : jTextField.setDocument(new JTextFieldLimitedDoc(nbMax, bDigit))
ActionAddPrintableAnnot	Action d'ajouter des annotation imprimables (traits de soulignement) dans le document PDF. Ces traits ne peuvent plus être effacés même en retirant les noms d'objet
ActionPref	Action d'ouverture de la boîte de dialogue des préférences
Menu	Fiche du menu principal de l'application DJIN
NewBibcodeDlg	Fiche de création d'un nouveau bibcode depuis le texte du presse-papier
ObjNameTransferHandler	Classe permettant le transfert par glisser-déplacer de noms d'objet entre la liste principale et les différentes listes des fenêtres enfant (rejetés, indéterminés, déjà traités)
ObjNameTransferable	Classe contenant une liste d'identificateurs à regrouper et implémentant l'interface java.awt.datatransfer.Transferable
PDFEdit	Programme de visualisation et modification d'un document PDF en utilisant les fonction de PDFBox. La modification ne fonctionne pas encore mais le programme permet d'afficher les groupes de caractères (ou positions) d'un document PDF avec leur coordonnées dans la page.
PDFElementDlg	Boîte de dialogue d'affichage des coordonnées, matrice de transformation et couleur d'un groupe de caractères cliqué dans le programme PDFEdit
PicturePanel	Composant graphique dérivé du JComponent de Swing permettant d'afficher une image
PrefDlg	Fenêtre des préférences de l'application DJIN
PropertiesDlg	Boîte de dialogue de saisie des propriétés d'un ou de plusieurs identificateurs
RegExpDlg	Boîte de dialogue permettant de d'inviter l'utilisateur à saisir une expression régulière à ajouter à l'arbre de recherche en lui proposant un historique des dernières expressions entrées.

RejectedDlg	Fenêtre d'affichage de la liste des noms d'objet rejetés (ou indéterminés ou déjà traités) dans une fenêtre enfant
SearchDlg	Fenêtre de recherche simple dans le texte extrait
SearchTableDlg	Boîte de dialogue de recherche avec préfixe dans une table et une colonne
TitleDjin	Fiche principale de TitleDjin (baptisé YesNo par Pascal Dubois) : programme de traitement des listes de titre en HTML
TransformDlg	Fenêtre effectuant les opérations chercher/remplacer sur les identificateurs
TreeRenderer	Classe de modification de l'affichage de l'arborescence. L'icône des éléments dépends du nombre de possibilités d'après le dictionnaire et la couleur dépend du résultat de la vérification du nom d'objet dans SIMBAD.
UnknownSymbolsDlg	Boîte de dialogue d'affichage des symboles graphiques non reconnus lors de l'extraction de texte. En cliquant dessus, l'utilisateur peut leur donner une signification qui ira s'ajouter à la liste d'images disponibles pour un journal donné.
VisuDlg	Boîte de dialogue permettant la visualisation d'un fichier ou d'un flux en temps réel. Permet d'afficher les fichiers de commandes de mise à jour SIMBAD ou le résultat de cette mise à jour pendant son exécution.
YesNoSimRef	Programme de lecture des titres du jour et ouverture d'une fenêtre YesNo sur les titres contenus dans le fichier sélectionné.

Figure 4. Diagramme des fenêtres du paquet Ihm



.1. Package cds.ObjectName.Ihm.Icon

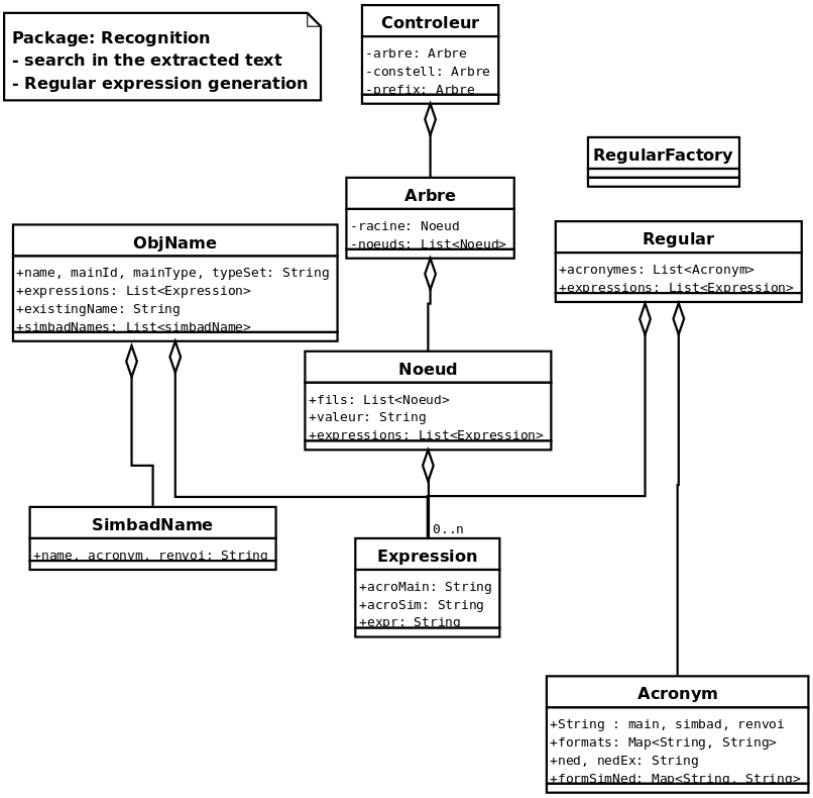
Icônes utilisées dans l'interface.

5. Package cds.ObjectName.Recognition

Table 4. Package cds.ObjectName.Recognition

Acronym	Acronyme d'un enregistrement du fichier main ou amas avec sa liste de formats SIMBAD et sa table de correspondance entre les formats SIMBAD et NED
Arbre	Arbre dont les noeuds sont des expressions régulières atomiques correspondant en général à un caractère. La toute première version de ce fichier provient du travail effectué par des stagiaires de l'ESIAL.
Controleur	Gestion des arbres de recherche et lancement des recherches de noms d'objet. Les fichiers contenant des listes de mots (suspects, adresses, lettres grecques, ...) sont également chargés par ce controleur. La toute première version de ce fichier provient du travail effectué par des stagiaires de l'ESIAL.
ExceptionArbre	Exception lors de la création d'un arbre. La toute première version de ce fichier provient du travail effectué par des stagiaires de l'ESIAL.
Expression	Expression régulière à chercher dans les documents. Contient les acronymes principal et Simbad.
MissingSemiColonException	Exception pour point-virgule (;) manquant dans les fichiers de BibCodes ou de résultat
Noeud	Représente un noeud d'un arbre. Chaque noeud contient une expression régulière d'un caractère, une référence au père et une liste de fils. La liste d'expressions régulières (avec acronyme) permet de retrouver les noms SIMBAD pouvant correspondre à un nom d'objet d'après le dictionnaire
NoMainAcronymException	Exception levée en cas d'absence de format principal dans le fichier main
RegularFactory	Classe de création de la liste des expressions régulières à partir du main, des amas, et des name. La fonction createList instancie l'objet Regular et lance les traitements pour lui faire écrire dans le fichier "liste" contenant les expressions régulières et qui sera ensuite lu pour générer l'arbre de recherche contenant les expressions régulières atomiques. Les sorties standard et erreur sont redirigées pour pouvoir afficher et garder traces des différents messages issus de cette génération.
Regular	Génération de la liste des expressions régulières à partir du dictionnaire de nomenclature. Ici sont effectués tous les traitements sur les fichiers (main, amas) du dictionnaire concernant les formats SIMBAD ou NED
Robot	Outil en ligne de commande permettant d'automatiser les traitements de DJIN sur des listes de documents ou les fichier parfile d'un répertoire ou afficher le numéro de version de DJIN
Validator	Traitement (pour test de validation ou autre) d'une liste de bibcode ou de document avec ou sans bibcode

Figure 5. Diagramme des classes du paquet Recognition

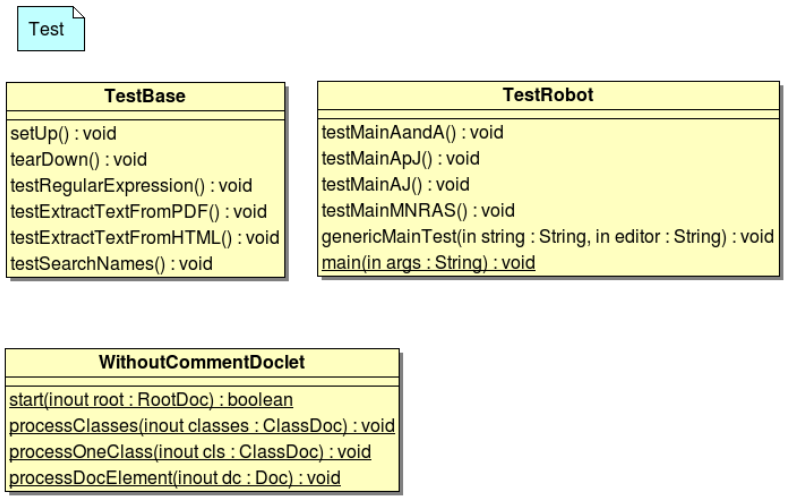


6. Package cds.ObjectName.Test

Table 5. Package cds.ObjectName.Test

TestBase	Test d'intégration pour certaines opérations de base : génération d'expressions régulièrs, extraction de texte depuis un documet PDF ou HTML, recherche de noms d'objet, lecture de fichier parfile
TestRobot	Test pour la validation de listes de documents avec vérification qu'une liste de noms d'objet donnés y ont bien été trouvés.
WithoutCommentDoclet	Doclet that prints out all method without comment of the class. Ce développement peut désormais être avantageusement remplacé par l'installation du plug-in d'Eclipse Checkstyle (http://eclipse-cs.sourceforge.net/)

Figure 6. Diagramme des classes du paquet Test

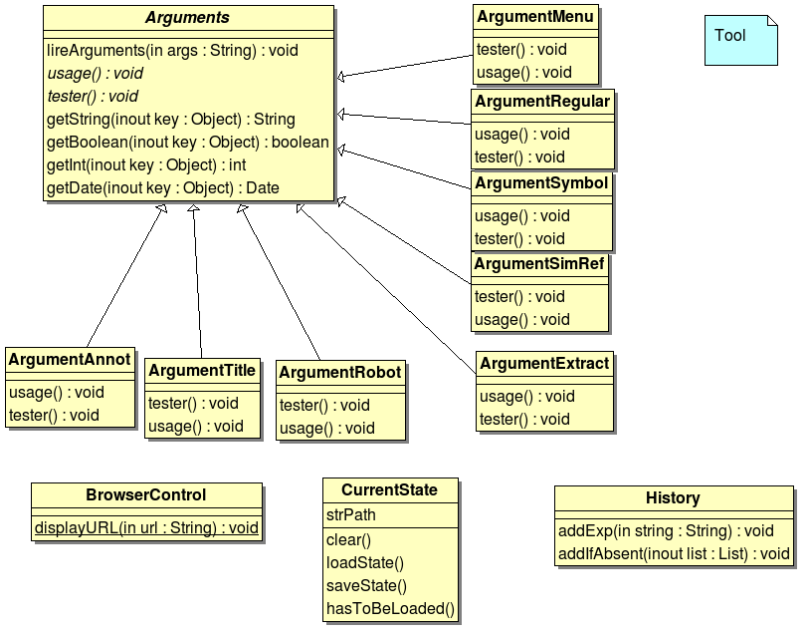


7. Package cds.ObjectName.Tool

Table 6. Package cds.ObjectName.Tool

ArgumentAnnot	Gestion des arguments du programme d'annotation d'un document PDF
ArgumentExtract	Arguments du programme d'extraction de texte
ArgumentFindCat	Gestion des arguments de la recherche dans les catalogues VizieR
ArgumentMenu	Arguments de l'IHM du menu principal
ArgumentNotes	Lecture des arguments de l'outil en ligne de commande NotesToHtml
ArgumentRegular	Gestion des arguments du programme de création de la liste des expressions régulières
ArgumentRobot	Lecture des arguments de l'outil en ligne de commande Robot
Arguments	Classe abstraite de lecture des arguments de la ligne de commande. Dériver cette classe pour chaque nouveau programme ayant des exigences spécifiques sur les arguments en implémentant la fonction "tester" qui vérifie la présence des arguments obligatoires et lance le cas échéant la fonction "usage" qui affiche la syntaxe du programme et quitte.
ArgumentSimRef	Arguments du programme YesNoSimRef
ArgumentSymbol	Arguments de l'utilitaire de comparaison d'images
ArgumentTitle	Arguments du programme de générations des titres en fichier parfile
BrowserControl	Classe pour l'affichage d'un fichier dans le navigateur du système
CurrentState	Classe permettant de sauvegarder et restaurer l'état courant du programme
History	Liste pour le stockage d'un historique d'expressions régulières

Figure 7. Diagramme des classes du paquet Tool



8. Package cds.vizier

Table 7. Package cds.vizier

CatName	Objet catalogue de VizieR (nom, vecteur d'occurrences et pertinence calculée)
DetailCatalog	Calcul du détail d'un score pour un catalogue unique et une recherche donnée
FindCatalog	Recherche de la liste des catalogues les plus pertinents pour une suite de mots donnée

9. Package cds.data.ident.ned

Table 8. Package cds.data.ident.ned

AcroFormat	Acronyme SIMBAD avec les formats NED et SIMBAD permettant d'écrire la table de correspondance entre les formats
Coordinate	Classe de manipulation de coordonnées dans les identificateurs
DicoNed	Génération à partir du dictionnaire de nomenclature (fichier main) de deux fichiers CSV séparés par des ';' Le premier contenant les champs des objets AcroFormat : Acronyme, format, expression régulière d'un identificateur SIMBAD, format d'un identificateur NED.
NedSimbad	Transformation d'un identificateur SIMBAD en identificateur NED
ReentrantTest	Test de traduction des identificateurs en mode multi-thread
SimbadNed	Transformation d'un identificateur SIMBAD en identificateur NED
test_SimbadToNed	test et démonstration d'utilisation de la fonction SimbadNed.simbadListToNed
TestElement	Elément du jeu de test de la transformation des identificateurs
ThreadTest	Thread de test de la traduction d'une liste d'identificateurs en mode multi-thread
TransformId	Transformation d'un identificateur d'une base dans une autre
TranslationTest	Programme de test de la transformation d'un jeu d'identificateur de SIMBAD vers NED

Informations sur ce document

Ce document est écrit au format DocBook[1]. C'est un format XML dont les balises servent à délimiter les différentes parties du contenu (titre, chapitre, paragraphe, liste, table, ...). Le contenu est ainsi séparé de la mise en forme.

Le fichier XML peut être transformé en HTML avec un fichier XSL qui se trouve dans l'installation Ubuntu standard, et aussi dans le paquet : **sudo apt-get install docbook-xsl**.

Transformation en HTML :

xsltproc -o djinDoc.htm /usr/share/apps/ksgmltools2/docbook/xsl/html/docbook.xsl djinDoc.xml

Le fichier XML peut être transformé en PDF avec le programme FOP[2]. La commande est la suivante :

~/objectname/fop/fop -param draft.mode no -param hyphenate false -param ulink.show 0 -xml djinDoc.xml -xsl /usr/share/xml/docbook/stylesheet/nwalsh/fo/docbook.xsl djindoc.pdf

Table 1. Signification des paramètres de la commande fop

Paramètre	Valeur	Signification
-param *nom*	*valeur*	Ces paramètres sont en fait destinés à la feuille de transformation XSL
-param draft.mode	no	Pas d'image de fond avec le mot "draft". Cette image était cherchée sur le site de sourceforge.net
-param hyphenate	false	La séparation des mots se trouve dans une bibliothèque séparée `fop-hyph.jar` pour un problème de licence, mais l'activation de ce mode fait planter le programme
-param ulink.show	0	Si la valeur est différente de 0, l'URL des liens externes est affichée à la suite entre crochet, ce qui n'est pas très esthétique
-param ulink.footnotes	1	Si la valeur est 1 et celle de *`ulink.show`* est 1, l'URL des liens externes est affichée en note de bas de page.
-param generate.toc	'article toc,title,figure,table'	Permet d'indiquer que l'on veut afficher le titre, la table des matières, des figures, des tables au début du document
-param toc.section.depth	2	Niveau de détail de la table des matières
-param section.autolabel	1	Numérotation automatique des sections
-xml	djinDoc.xml	Contenu de la documentation au format XML
-xsl	docbook.xsl	Fichier XSL de transformation vers le format XSL-FO utilisé ensuite par FOP pour générer le PDF

Pour que les liens internes apparaissent soulignés dans le document PDF, l'argument *`text-decoration="underline"`* doit être ajouté à la ligne 134 du fichier `/usr/share/xml/docbook/stylesheet/nwalsh/fo/inline.xsl` : *`<fo:basic-link text-decoration="underline" internal-destination="{$linkend}">`*

Le fichier XML étant lié à une feuille de style CSS, il peut être visualisé directement dans un navigateur.

OpenOffice peut lire le format DocBook directement.

[1] http://www.docbook.org
[2] http://xmlgraphics.apache.org/fop/index.html

L'éditeur XXE[3] (XMLmind XML Editor) permet d'éditer le fichier XML source en masquant les balises mais en les gardant tout de même accessibles. De plus, le fichier XML produit est indenté proprement. Ce logiciel est gratuit pour une utilisation personnelle ou pour créer des documents sous licence open source.

D'autres fichiers XSL existent qui permettent la transformation dans d'autres formats (LaTeX, man page, HTML Help, ...).

Documentation des différentes balises DocBook[4]

Seul un petit nombre de ces balises a été utilisé dans ce document. En voici la liste :

1. Balises Docbook

1.1. Structure du document

article[5], book[6], index[7], para[8], sect1[9], sect2[10], sect3[11], sect4[12], sect5[13], title[14]

1.2. Mise en forme

`classname` [15], *emphasis* [16], `filename` [17], **keycap** [18], note[19], *`parameter`* [20],

```
programlisting
```

[21], “quote” [22], superscript [23], tip[24], warning[25]

1.3. Renvois

anchor[26], indexterm[27], link[28], ulink[29],

1.4. Listes

itemizedlist[30], listitem[31], orderedlist[32]

[3] http://www.xmlmind.com/xmleditor/
[4] http://docbook.org/tdg/en/html/quickref.html
[5] http://docbook.org/tdg/en/html/article.html
[6] http://docbook.org/tdg/en/html/book.html
[7] http://docbook.org/tdg/en/html/index.html
[8] http://docbook.org/tdg/en/html/para.html
[9] http://docbook.org/tdg/en/html/sect1.html
[10] http://docbook.org/tdg/en/html/sect2.html
[11] http://docbook.org/tdg/en/html/sect3.html
[12] http://docbook.org/tdg/en/html/sect4.html
[13] http://docbook.org/tdg/en/html/sect5.html
[14] http://docbook.org/tdg/en/html/title.html
[15] http://docbook.org/tdg/en/html/classname.html
[16] http://docbook.org/tdg/en/html/emphasis.html
[17] http://docbook.org/tdg/en/html/filename.html
[18] http://docbook.org/tdg/en/html/keycap.html
[19] http://docbook.org/tdg/en/html/note.html
[20] http://docbook.org/tdg/en/html/parameter.html
[21] http://docbook.org/tdg/en/html/programlisting.html
[22] http://docbook.org/tdg/en/html/quote.html
[23] http://docbook.org/tdg/en/html/superscript.html
[24] http://docbook.org/tdg/en/html/tip.html
[25] http://docbook.org/tdg/en/html/warning.html
[26] http://docbook.org/tdg/en/html/anchor.html
[27] http://docbook.org/tdg/en/html/indexterm.html
[28] http://docbook.org/tdg/en/html/link.html
[29] http://docbook.org/tdg/en/html/ulink.html
[30] http://docbook.org/tdg/en/html/itemizedlist.html

1.5. Tables

colspec[33], entry[34], informaltable[35], row[36], spanspec[37], table[38], tbody[39], thead[40], tgroup[41]

1.6. Figures

figure[42], graphic[43], inlinegraphic[44]

[31] http://docbook.org/tdg/en/html/listitem.html
[32] http://docbook.org/tdg/en/html/orderedlist.html
[33] http://docbook.org/tdg/en/html/colspec.html
[34] http://docbook.org/tdg/en/html/entry.html
[35] http://docbook.org/tdg/en/html/informaltable.html
[36] http://docbook.org/tdg/en/html/row.html
[37] http://docbook.org/tdg/en/html/spanspec.html
[38] http://docbook.org/tdg/en/html/table.html
[39] http://docbook.org/tdg/en/html/tbody.html
[40] http://docbook.org/tdg/en/html/thead.html
[41] http://docbook.org/tdg/en/html/tgroup.html
[42] http://docbook.org/tdg/en/html/figure.html
[43] http://docbook.org/tdg/en/html/graphic.html
[44] http://docbook.org/tdg/en/html/inlinegraphic.html

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Y