Agent d’extraction

Classe de l’agent: com.clog.workey.agents.extract.ExtractionAgent

Cet agent permet l’extraction des documents et de leurs pièces-jointes ayant été modifiés depuis sa dernière exécution.

Par défaut, les documents sont exportés en XML, sous la même forme que l’archiveur ou l’API Rest. Les fichiers XML extraits sont nommés document-III-yyyyMMddHHmm.xml, où III est l’identifiant du document, suivi du timestamp de l’agent d’exportation. S’il existait un précédent document avec le même identifiant dans le répertoire il sera supprimé, de même que pour le répertoire des pièces-jointes.

Il est également possible d’exécuter d’autres types de traitement sur ces documents à l’aide de classes Java ou de scripts Groovy.

Installation

Le fichier workey-agent-extraction-x.y.z.jar (où x.y.z est le numéro de version) doit être placé dans le répertoire $TOMCAT_HOME/workey/libs.

Paramètres

- **outputDirectory** : Répertoire où seront exportés les documents et les pièces jointes. Ce paramètre est obligatoire si le processeur par défaut est appelé (voir plus bas).
- **processName** : Nom interne du processus à traiter. Si cette valeur est absente, tous les documents, de tous les processus sont pris en compte.
- **doctypeName** : Nom interne du type de document à traiter. Si ce paramètre est rempli, il est nécessaire d’avoir aussi défini le nom du processus.
- **workflowManager** (*obligatoire*) : Identifiant du gestionnaire de workflow qui va extraire les données.
- **processorClass** : Nom d’une classe Java implémentant l’interface com.clog.workey.agents.extract.ExtractionDocumentProcessor qui autorise un traitement supplémentaire lors de l’export de chaque document et pièce-jointe. Ce paramètre permet de faciliter l’intégration avec des moteurs de reporting. Plusieurs noms de classes peuvent être indiqués, séparés par les caractères suivants : “;“, “:” ou “,“. Voir ci-dessous.
- **exportAttachments** : Si la valeur de cette propriété est “true“, les pièces-jointes seront extraites.
- **dateRange** : Si cette option est activée, le mode de sélection de l’agent n’est plus le mode normal (export des documents modifiés depuis son dernier lancement). Les documents compris entre les dates définies dans ce paramètre seront exportés. Le format de ce paramètre est yyyyMMddHHmm-yyyyMMddHHmm. Cette option peut être utile pour faire une exportation globale des documents déjà présent sur le serveur.

N.B.: si cette option est utilisée, la date d'exécution de l'agent n'est pas mise à jour afin de ne pas fausser des extractions ultérieures en mode "normal".

Classes de traitement des documents

Traitement par défaut

Export des documents

Afin d'éviter de se retrouver avec un trop grand nombre de fichiers par répertoire, les fichiers sont uniformément distribués dans 2*255 sous-niveaux de répertoire (de 00/ à ff/). Si les pièces-jointes sont exportées, elles seront toujours dans la même branche que leur document. Ainsi, un document et ses pièces-jointes seront par exemple exportés ainsi :

```
+ outputDirectory
  +-37
  | +-d2
  | | +-1234
  | | | +-document-1234-202102101145.xml
  |     +-document-1234-attachments/
          +-297e458a76d8ae8c0176d8c0772d0001-fichier.docx
          +-297e458a76f672490176f69c670b000b-facture.pdf
```

N.B. les fichiers sont sauvegardés sur disque de cette manière seulement si le processeur par défaut est appelé (dans les cas où le paramètre processorClass est vide, ou bien s'il contient la valeur <D>).

Export des pièces-jointes

Les pièces jointes sont copiées dans un sous-répertoire nommé "document-{docId}-attachments/". Le nom des fichiers extrait est constitué de l'identifiant unique de la pièce-jointe (*uuid*) et de son nom original.

Appel d'autres classes de traitement

Pour combiner cette classe avec d'autres classes de traitement, vous pouvez utiliser son nom raccourci "<D>" dans le paramètre processorClass. Également, si le nom de la classe commence par "*P*", le package com.clog.workey.agents sera utilisé.

Par exemple pour sauvegarder les fichier et créer un fichier Excel, vous pouvez mettre comme valeur : <D>;*P*.ExcelExtractionProcessor (qui est équivalent à com.clog.workey.agents.extract.DefaultExtractionProcessor;com.clog.workey.agents.extract.ExcelExtractionProcessor, mais en plus succinct).

À l'instar des connecteurs Workey, si des classes de traitements externes sont développées, le fichier .jar contenant ces classes doit être placé dans le répertoire $TOMCAT_HOME/workey/libs.

ExcelExtractionProcessor – Récapitulatif Excel

- Classe “com.clog.workey.agents.extract.ExcelExtractionProcessor” : Créé un fichier Excel récapitulant les données extraites lors de l’exécution de l’agent, avec une feuille pour les documents exportés et une autre pour les pièces-jointes.

GroovyExtractionProcessor – Exécution d’un processor par script groovy

- Classe “com.clog.workey.agents.extract.GroovyExtractionProcessor” : Charge un script nommé ExtractionGroovy.groovy dans le répertoire de sortie de l’agent, ou le script indiqué par la propriété système com.clog.workey.agents.extract.GroovyExtractionProcessor.script. Ce script doit comporter les méthodes suivantes pour fonctionner correctement. Elles seront appelées de la même manière qu’une classe java implémentant l’interface ExtractionDocumentProcessor (voir ci-dessous).

```
void initDocumentProcessor(extractionAgent) {
    /* a remplir */
}

void processDocument(documentId, xmlDoc) {
    /* a remplir */
}

void processAttachment(documentId, xmlDoc, attachmentElement, attachment,
data) {
    /* a remplir */
}

void closeDocumentProcessor() {
    /* a remplir */
}

boolean filterDocument(documentId, xmlDoc) {
    /* a remplir */
}
```

Voici un autre exemple d’un script insérant des données des documents et des pièces-jointes dans une base de données externe :

```
import javax.naming.InitialContext;
import groovy.sql.Sql;
import java.sql.Timestamp;
import java.util.Date;
```

```
import java.text.SimpleDateFormat;

datasource = null;
sql = null;
startTime = null;

/* Exemple de schema mySQL:
 * CREATE TABLE attachments (
 *   uuid char(32) NOT NULL,
 *   doc_id bigint(20) NOT NULL,
 *   file_name text NOT NULL,
 *   extract_time datetime NOT NULL
 * );
 * ALTER TABLE attachments ADD PRIMARY KEY (uuid);
 *
 * CREATE TABLE documents (
 *   id bigint(20) NOT NULL,
 *   current_state text NOT NULL,
 *   title text NOT NULL,
 *   last_modified datetime NOT NULL,
 *   last_actor text NOT NULL,
 *   extract_time datetime DEFAULT NULL
 * );
 * ALTER TABLE documents ADD PRIMARY KEY (id);
 */

void initDocumentProcessor(extractionAgent) {
    startTime = System.currentTimeMillis();

    // Utilisation de la datasource Workey:
    //datasource = new
InitialContext().lookup('java:comp/env/WorkeyReportDS');
    //sql = Sql.newInstance(datasource);

    // ou bien d'une connexion directe:
    Map dbConnParams = [
        url: 'jdbc:mysql://localhost/workey-report',
        user: 'root',
        password: '',
        driver: 'com.mysql.jdbc.Driver'
    ];
    sql = Sql.newInstance(dbConnParams);
}

void processDocument(documentId, xmlDoc) {
    sdfTime = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS");

    // Extraction des données par XPath
    def state = xmlDoc.selectSingleNode(
        "/wky:document/wky:header/wky:state/@name").text;
    def subject = xmlDoc.selectSingleNode(
```

```
        "/wky:document/wky:header/wky:subject").text;
    def updateTime = new Timestamp(sdfTime.parse(xmlDoc.selectSingleNode(
        "/wky:document/@update-time").text).time);
    def actor = xmlDoc.selectSingleNode(
        "//wky:history/wky:stamp[position()=last()]/wky:actor/@name").text;
    def extractTime = new Timestamp(System.currentTimeMillis());

    // Insertion dans la base de données
    sql.execute """
        REPLACE INTO documents
            (id, current_state, title, last_modified,
             last_actor, extract_time)
        VALUES (${documentId}, ${state}, ${subject}, ${updateTime},
             ${actor}, ${extractTime})
    """;
}

void processAttachment(documentId, xmlDoc, attachmentElement, attachment,
data) {
    def uuid = attachment.UUID;
    def fileName = attachment.fileName;
    def extractTime = new Timestamp(System.currentTimeMillis());

    // Insertion dans la base de données des données de
    // la pièce-jointe.
    sql.execute """
        REPLACE INTO attachments
            (uuid, doc_id, file_name, extract_time)
            VALUES (${uuid}, ${documentId}, ${fileName}, ${extractTime})
    """;
}

void closeDocumentProcessor() {
    sql.close();
    def timeInSec = (System.currentTimeMillis() - startTime) / 1000;
    println("Groovy process done in ${timeInSec} sec.");
}

boolean filterDocument(documentId, xmlDoc) {
    return true;
}
```

Ce script existe également sous forme de classe Java “com.clog.workey.agents.extract.DatabaseExtractionProcessor“.

StarpageStart – Démarrage Starpage

Classe “com.clog.workey.agents.extract.StarpageStart” : création d’un fichier “start” en fin de traitement de l’agent pour le démarrage automatique du *daemon* Starpage.

Interface ExtractionDocumentProcessor

Cette interface est composée de cinq méthodes :

- initDocumentProcessor(ExtractionAgent extractionAgent): Elle est exécutée avant le traitement des documents. Une instance de l'agent est passée en paramètre, qui permet d'accéder au répertoire de sortie (getBaseOutputDirectory()), au timestamp utilisé par cet agent (getAgentTimestamp()), aux noms du processus et du type de document le cas échéant (getProcessName() et getDocumentTypeName()).
- filterDocument(long docId, Document xmlDoc) : Cette méthode est appelée après chaque ouverture de document pour savoir s'il doit être inclus dans l'export. Doit renvoyer "true" si le document doit être exporté, "false" s'il ne doit pas l'être.
- processDocument(long docId, Document xmlDoc) : Cette méthode est appelée après l'export de chaque document.
- processAttachment(long docId, Document xmlDoc, Element attachmentElement, DocumentData attachment, InputStream data) : Cette méthode est appelée après l'export de chaque pièce-jointe.
- closeDocumentProcessor() : Cette méthode est appelé à la fin de l'exécution de l'agent.

De plus, la classe ExtractionAgent dispose d'un méthode Map<String, Object> getExtractionContext() qui permet d'échanger des informations entre les différentes classes de traitement. Ainsi, la classe de traitement par défaut inclura le nom du document en cours de traitement dans la clé DefaultExtractionProcessor.XML_OUTPUT_FILE, et les fichiers des pièces-jointes sauvegardés dans les clés avec le UUID de la pièce-jointe.

Par exemple une classe minimaliste de traitement pourrait ressemble à :

```
package com.clog.workey.agents.extract;

import java.io.InputStream;

import org.dom4j.Document;
import org.dom4j.Element;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.clog.workey.core.DocumentAttachment;

public class DummyExtractionProcessor implements ExtractionDocumentProcessor
{
    private static final Logger logger =
LoggerFactory.getLogger(DummyExtractionProcessor.class);

    @Override
    public void initDocumentProcessor(ExtractionAgent extractionAgent) throws
```

```
ExtractionProcessorException {
        logger.info("initDocumentProcessor");
    }

    @Override
    public void processDocument(long documentId, Document xmlDoc) throws
ExtractionProcessorException {
        logger.info(String.format("processDocument %d", documentId));
    }

    @Override
    public void processAttachment(long documentId, Document xmlDoc, Element
attachmentElement,
            DocumentAttachment attachment, InputStream data) throws
ExtractionProcessorException {
        logger.info(String.format("processAttachment %s from doc %d",
attachment.getUUID(), documentId));
    }

    @Override
    public void closeDocumentProcessor() throws ExtractionProcessorException
{
        logger.info("closeDocumentProcessor");
    }

    @Override
    public boolean filterDocument(long documentId, Document xmlDoc) throws
ExtractionProcessorException {
        logger.info(String.format("filterDocument %d", documentId));
        return true;
    }
}
```