Connecteur Conversion .docx vers .pdf

Connecteur

com.clog.workey.connectors.DocxToPdfConnector

Action

Transformation de fichiers docx contenus dans un champ pièce-jointe en pdf.

En entrée

1. Champ de type pièce-jointe. Peut être un champ multivalué.

En sortie :

1. Liste des UUID des pièces-jointes converties en PDF. Peut servir à alimenter un champ pièce-jointe.

Statistiques MixPanel

- Auteur de l'article

Par Briac Date de l'article

- 2021-04-19

Lorsque la fonctionnalité de suivi MixPanel est activée, certaines actions du moteur vont alimenter de manière anonyme la base d'événement MixPanel.

Attention

L'intégration de MixPanel n'est disponible que pour les version **6.11.4** de Workey et suivantes.

Afin de minimiser l'impact de la collecte de ces événements, ils sont envoyés dans une queue JMS qui les traitera au fur et à mesure de manière asynchrone. Cependant, il n'est pas garanti que 100% des messages devant générer un événement MixPanel soient pris en compte.

Installation

Par défaut, cette fonctionnalité est désactivé. Pour la mettre en place, il suffit d'ajouter la propriété suivante au fichier $TOMCAT_HOME/conf/catalina.properties :

com.clog.workey.analytics.mixpanel.enable=true

Évènements suivis

- **Create Document** : Déclenché lors de la demande de création d'un nouveau document. En plus des propriétés standard, il contient les propriétés workey-process, workey-document-type, workey-state et workey-role.
- Open Document : (non suivi car peu fiable)
- **Submit Document** : Déclenché à la fin de la soumission d'un document (donc pas en cas d'erreur). En plus des propriétés standard, il contient les propriétés :
  - workey-document,
  - workey-process,
  - workey-document-type,
  - workey-state,
  - workey-role,
  - workey-agent, booléen indiquant si la soumission a été faite par un agent
  - workey-document-alive, booléen indiquant si le document est arrivé en fin de vie (plus d'état suivant disponible)
- **Save Document** : Déclenché lors de l'enregistrement d'un document (sans changement d'état). En plus des propriétés standard, il contient les propriétés workey-document, workey-process, workey-document-type, workey-state et workey-role.
- **Open View** : Déclenché lors de l'ouverture ou rafraîchissement d'une vue. En plus des propriétés standard, il contient les propriétés workey-view.
- Error during submit : (non suivi actuellement)
- **Execute Search** : Déclenché lorsque une recherche est effectué. Aucune propriété particulière n'est envoyée.
- **Deploy Process** : Déclenché lorsqu'un processus est déployé. Pour des raisons techniques, l'utilisateur effectuant ce déploiement ne peut pas être distingué, un utilisateur avec un id "-1" et unique pour la license est utilisé. En plus des propriétés standard, il contient les propriétés :
  - workey-process-name
  - workey-deploy-roles, le nombre de rôles déployés dans ce processus
  - workey-deploy-operations, le nombre d'opérations déployées dans ce processus
  - workey-deploy-doctypes, le nombre de types de document déployés dans ce processus
  - workey-deploy-forms, le nombre de formulaires déployés dans ce processus
  - workey-deploy-views, le nombre de vues déployées dans ce processus

Propriétés MixPanel

Pour des raisons de sécurités, toutes les propriété ci-dessus (hormis les champ booléen) sont pseudonymisées à partir de la licence.

Cette pseudonymisation prend la forme d'un UUID créé à partir du nom du client, du type de données (document-id, operation-id, etc.) et de l'identifiant de la donnée. Cela permet d'identifier de manière unique chaque partie de données sans avoir les données réelles.

Lors de chaque événement envoyé à MixPanel, les propriétés suivantes sont également envoyées :

- workey-client : le nom du client (champ “Licensed To” de workey-license.xml), c’est la seule donnée en clair concernant le client.
- workey-version : Version de Workey utilisée pour cet événement (“6.11.4“, par exemple)
- workey-env : Valeur (en clair) de la propriété système “com.clog.workey.analytics.mixpanel.env“, si elle est présente. Cela permet de filtrer certaines installations particulières, si besoin

Propriétés Workey

- com.clog.workey.analytics.mixpanel.enable : Activation/désactivation de l’envoi d’événements vers MixPanel (“true” ou “false“, “false” par défaut)
- com.clog.workey.analytics.mixpanel.token : Identifiant de l’environnement MixPanel utilisé (“1c1358cdd54edfb2341e98682f7956e2“: production (par défaut), “d714004bdc15c44bffa9313eea27f60f“: test)
- com.clog.workey.analytics.mixpanel.env : Propriété système alimentant la propriété MixPanel “workey-env” (voi ci-dessus)

Connecteur CSV Sage

- Auteur de l’article

Par Briac Date de l’article

- 2021-03-22

Connecteur permettant la création d’un fichier CSV pour Sage comportant des informations relatives à une facture.

Format du fichier CSV

Le fichier CSV produit par ce connecteur contient les champs dans l’ordre de la table ci-dessous. Le caractère séparateur de champ est “;“, sans échappement des chaînes de caractères. Les nombres sont au format français, sans séparateur des milliers (“1234,56” par ex).

Champ CSV	Description
Nom du journal	Nom du journal à utiliser (“ACH“, par ex.)
Date de la facture	Date de la facture au format YYYYMMDD
Date d’échéance	Date d’échéance de la facture au format YYYYMMDD
Numéro chronologique	Identifiant unique (Id du document Workey, par ex)
Numéro de la facture	Numéro indiqué sur la facture
Compte à imputer	Numéro d’imputation comptable

Champ CSV	Description
Code fournisseur	Identifiant du fournisseur (présent seulement sur la première ligne)
Libellé de l’écriture	Libellé de l’écriture
Montant TTC	Montant TTC de l’entrée
Débit/Crédit	Indique si l’entrée est une opération de débit ou crédit (“D” ou “C“)
Moyen de paiement	Code du moyen de paiement (entier)
Devise	Code de la devise (“EUR“, par ex)
Cours de la devise	Cours de la devise (“1” si aucune conversion n’est nécessaire)
Total devise	Résultat de l’opération Montant TTC * Cours devise
Bon à payer	Indique si la facture est validée pour paiement (“OUI” ou “NON“)

Champs CSV exportés par le connecteur

Appel du connecteur

Classe : com.clog.workey.connectors.sage.ExportSage

Paramètres d’entrée

1. Champ ayant comme valeur un objet JSON comportant la configuration des différents champs à utiliser.

Valeurs de sortie

1. (*Aucune*)

Configuration

Propriétés systèmes

Plusieurs propriétés systèmes peuvent être configurées dans le fichier $TOMCAT/conf/catalina.properties pour définir des valeurs par défaut à utiliser :

- com.clog.workey.connectors.ExportSage.exportDir : Répertoire où seront créés les fichiers CVS pour Sage (par défaut $TOMCAT/workey/sage/)
- com.clog.workey.connectors.ExportSage.journal : Nom par défaut du journal si aucun champ journal n’est présent dans le formulaire (“ACH“, par ex).
- com.clog.workey.connectors.ExportSage.compteTiers : Compte tiers par défaut si aucun champ n’est présent dans le formulaire.

Configuration JSON

La configuration permet d’effectuer un “mapping” entre les champs présents dans le formulaire Workey et les champs attendus dans le CVS Sage. Pour

chaque propriété, il est possible d'indiquer le nom du champ qui lui correspond. Sinon, le nom de champ par défaut et/ou la valeur par défaut sera utilisé.

```
{
  "codeFournisseur": "Fournisseur",
  "compteGeneral": "Compte_debit",
  "devise": "Monnaie"
}
```

Liste des propriétés

Propriété	Nom du champ par défaut	Observations
journal	Journal	propriété *com.clog.workey.connectors.ExportSage.journal*
compteTiers	Compte_tiers	propriété *com.clog.workey.connectors.ExportSage.compteTiers*
compteGeneral	Compte_general	
dateFacture	Date_facture	
dateEcheance	Date_echeance	
numeroFacture	Numero_facture	
codeFournisseur	Code_fournisseur	
libelleEcriture	Libelle_ecriture	
montant	Montant_TTC	valeur par défaut : 0
detailMontant	Montant_net	champ multivalué
detailCompte	Compte_general	champ multivalué
detailLibelle	Libelle	champ multivalué
modePaiement	Moyen_paiement	
devise	Devise	valeur par défaut : EUR
coursDevise	Cours_devise	valeur par défaut : 1
bonAPayer	Bon_a_payer	valeur par défaut : false
tvaIntracommunautaire	TVA_Intracommunautaire	valeur par défaut : false
compteTvaIntraCredit	Compte_TVA_intra_credit	
compteTvaIntraDebit	Compte_TVA_intra_debit	
typeFacture	Type_facture	valeurs possibles : INVOICE (facture) ou CREDIT NOTE (avoir)

Propriétés pour la correspondance Workey/CSV

En plus de ces propriétés utilisées pour la correspondance entre les champs Workey et les champs CSV, il est possible de redéfinir le nom du fichier de sortie avec la propriété "fileTemplate". Sa valeur par défaut est "ExportSage-%s.csv", où le "%s" représente la date (au format YYYYMMDD). Par conséquent, un seul fichier d'export par jour est créé, qui va contenir toutes les factures traitées dans la journée.

Attention

Si la propriété "fileTemplate" est redéfinie, la marque "%s" doit toujours être présentes.

TVA Intracommunautaire

Si la propriété "tvaIntracommunautaire" est à "true", deux lignes

supplémentaires seront créés dans le fichiers CSV : l'une pour l'opération de crédit avec le compte d'imputation indiqué par la propriété "compteTvaIntraCredit" et l

Limitations

Actuellement, la ventilation de la facture sur plusieurs compte n'est pas supportée.GitbucketICescrum

- Étiquettes
- Connecteur

Workey Selenium Tester

- Auteur de l'article

Par Briac Date de l'article

- 2021-03-19

Table of contents

Cet article présente l'installation, la configuration et l'utilisation de la plateforme Workey d'automatisation des tests "workey-selenium-tester".

Installation

- Télécharger la dernière version dans GitBucket: http://jawa.c-log.lan:8087/workey/workey-selenium-tester/releases
- Dézipper le ficher workey-selenium-tester-1.5.1.zip dans un répertoire

de votre système.
- Si besoin, importer le fichier des utilisateurs de tests resources\ldif\test-users.ldif dans le LDAP (les tests de validation s'attendent à trouver certains de ces utilisateurs).
- Éditer si besoin les fichiers admin-rest.properties et validation-selenium.properties

Scripts Groovy

Les scripts et WKY de validation Workey se trouvent dans le répertoire tests/validation. Le script tests/validation/000-setup.groovy est particulier car c'est lui qui déploie les WKY, créé les unités organisationnelles et effectue les assignations des acteurs à leur rôles. Il prend ses informations des fichiers tests/validation/000-assignments.json et tests/validation/000-organizational_units.xml

Plusieurs scripts Groovy ont été développés pour automatiser la création de formulaires, ordres et demandes dans SIRA. Ils se trouvent dans le répertoire groovy/. Les scripts utilisés pour tester le processus de blocage-v2 se trouvent dans le répertoire test/blocage-v2.

Les scripts Groovy présents dans le répertoire test/validation/ sont numérotés en essayant de respecter la convention arbitraire :

- 000: Administration (spécial)
- 001 à 599 : Fonctionnalités de base de Workey
- 600 à 899 : Fonctions spéciales / modules / etc.
- 900 à 999 : Connecteurs

Dans l'idéal, à chaque script Groovy doit correspondre un projet Workey associé avec le même nom (seule l'extension change).

Dans les cartons…

Il faudrait faire en sorte (avec une sorte de fichier Manifest) que le script Groovy puisse déployer lui-même le processus et faire les affectations avant de se lancer.

Lancement

Les options de workey-selenium-tester:

```
usage: workey-selenium-tester [options] scripts.groovy...
 -l,--loop <LOOP>         Number of time each groovy script has to be
executed
 -i,--input-file <FILE>   run tests found in FILE
 -t,--type <TYPE>         runner type WORKEY5, WORKEY6 or WORKEY7 (default
WORKEY6)
 -5,--workey5             Set the runner as Workey 5
 -6,--workey6             Set the runner as Workey 6 (default)
 -7,--workey7             Set the runner as Workey 7
 -v,--verbose             Prints more message from Selenium driver
```

```
 -h,--help                Prints this help message and exit
```

Pour exécuter un script Groovy:

```
.\bin\workey-selenium-tester.bat .\tests\blocage-v2\test00-treeview.groovy
```

Pour exécuter tous les tests contenus dans un fichier contenant un fichier .groovy par ligne :

```
.\bin\workey-selenium-tester.bat -i tests_workey.list
```

Une fois tous les scripts exécutés, les statistiques de réussite et d'échec sont affichées. Si des erreurs sont survenues, la liste des scripts problématiques est enregistrées dans le fichier "failed-scripts.list".

Dans PowerShell, pour purger les processus des navigateurs robots :

```
stop-process -name chromedriver
```

Dans les cartons…

Prévoir un GUI pour faciliter le lancement des tests et pouvoir enregistrer le résultat des tests de manière synthétique (CSV, XML ou JSON).

Compilation

- `.\gradlew.bat installZip`
- `.\build\install\workey-selenium-tester\bin\workey-selenium-tester.bat .\tests\blocage-v2\test00-treeview.groovy`

ou bien,

- `.\gradlew.bat distZip`

Écriture de tests Workey Selenium

Configuration

Fichier de Propriétés

Le constructeur du robot Selenium prend en paramètre un chemin vers un fichier de propriétés. Les différentes propriétés possibles sont les suivantes :

- selenium.driver.host : URL de connexion vers le serveur Workey
- selenium.driver.type : Type de navigateur à utiliser: "chrome" (par défaut), "firefox" ou "edge". Le driver Chrome est celui qui est privilégié.
- selenium.driver.timeout : Temps d'attente en seconde lors de la recherche d'éléments.

Propriétés spécifiques Chrome

- selenium.driver.chrome.profile : Réutilise le profil Chrome au lieu d'en créer un nouveau à chaque exécution. Cela permet notamment de réutiliser des extensions (valeur "true"/"false", défaut à "false").
- selenium.driver.chrome.headless : Lance Chrome sans interface graphiqe (valeur "true"/"false", défaut à "false").
- selenium.driver.chrome.allow-notifications : Autorise les notifications à s'afficher (valeur "true"/"false", défaut à "false").
- selenium.driver.chrome.maximized : Lance Chrome avec la fenêtre maximisée (valeur "true"/"false", défaut à "false").
- selenium.driver.chrome.disable-gpu : Lance Chrome sans accélération graphique, permet parfois de corriger des bugs visuels (valeur "true"/"false", défaut à "false").
- selenium.driver.chrome.disable-infobars : Désactive l'affichage d'une barre indiquant que Chrome est lancé en mode automatique (valeur "true"/"false", défaut à "false").

Propriétés avancées

- selenium.driver.skip-pause : Ignore les pauses manuelles. Utile lorsque les tests doivent être complètement automatiques (valeur "true"/"false", défaut à "false").
- screenshot.dir : Répertoire où seront enregistré les copies d'écran (par défaut "output")
- selenium.proxy : Créé un proxy permettant d'enregistrer les communications au format HAR avec les méthodes runner.newHar(harFile) et runner.saveHar() (valeur "true"/"false", défaut à "false").
- selenium.driver.path : Chemin d'accès aux drivers des navigateurs (par défaut "drivers")
- test.language : Code du langage utilisé pour les données créées par JFairy (par défaut "fr")
- runner.type : Type de robot à utiliser (valeur "WORKEY_5", "WORKEY_6" ou "WORKEY_7")

Déploiement et initialisation des processus

Le script "validation/000-setup.groovy" est un script spécial qui permet de définir les gestionnaires de workflow et de processus, de déployer automatiquement des fichiers WKY, de créer des unités organisationnelles et d'affecter des acteurs à des rôles et à des unités.

Il prend ces informations à partir du fichier "admin-rest.properties" dans lequel est défini un chemin vers un fichier JSON (voir par exemple, "tests/validation/000-assignments.json")

Dans les cartons…

Cette partie va vraisemblablement être modifiée pour être un peu plus flexible.

Script Groovy

Le “Runner” Workey

L’application dispose de plusieurs robots spécialisés dans différentes versions de Workey. Ils peuvent être forcés avec l’option “--type” de workey-selenium-tester.

- com.workey.selenium.SeleniumScriptRunner : Workey 5 / JBoss
- com.workey.selenium.SiraScriptRunner : Workey 5 / JBoss / Fonctions spéciales SIRA (Gencods, création des formulaires, etc.)
- com.workey.selenium.Workey6ScriptRunner : Workey 6 / Tomcat
- com.workey.selenium.WorkeyStoreScriptRunner : Workey 7 – nouvelle interface/ Tomcat

Principales propriétés

- runner : le “robot” Workey, couche de haut niveau qui sait se connecter à l’application, ouvrir des vues, créer des document, remplir des formulaires, etc
- runner.driver : le moteur Selenium, pour effectuer des actions de bas niveau qui ne seraient pas proposé par le runner.
- runner.logger : instance de logger
- runner.fairy : Générateur de données factices mais vraisemblables (nom de personne, adresses, société, mail, etc). Pour plus d’informations: JFairy.
- runner.textProducer : Générateur de texte aléatoire
- runner.properties : Propriétés chargées lors de l’exécution du script

Principales méthodes Workey

- runner.getCredentials("XXX") : récupère le login/mot de passe à partir des propriétés systèmes “XXX.login” et “XXX.password“. Si ces propriétés n’existent pas, le login “XXX” est utilisé, avec le mot de passe “test“.
- runner.workeyLogin(Credentials) : Connexion à l’application.

```
// Connexion à workey avec l'utilisateur test001
runner.workeyLogin(new Credentials('test001', 'pa$$w0rd'));
```

- runner.workeyLogout() : Déconnexion
- runner.openView(viewName) : Ouverture d’une vue (par son libellé en v6 et v7, par son nom interne en v5).

```
// Ouverture de la vue "A Faire" en v6
runner.openView('A Faire')
```

- runner.openDocumentFromView(documentIdentifier) : Ouverture d’un document depuis une vue. Le paramètre doit être de préférence une chaîne de caractère permettant d’identifier sans ambigüité le document à ouvrir.

```
// Ouverture d'un document depuis la vue courante
runner.openDocumentFromView('DocumentTest-1C5-FKI-5CT')
```

- runner.createNewDocument(documentLabel) : Création d'un nouveau document, en indiquant le libellé avec lequel il apparaît dans le menu de création.

```
runner.createNewDocument('Demande de congés')
runner.createNewDocument('Note de frais [Valideur]')
```

- runner.setFieldValue(fieldName, value): Change la valeur d'un champ.

Attention

Le type de la donnée "value" doit être une chaîne de caractère.

- runner.setFieldValue(fieldName, value, index): Si le champ est multivalué, ou s'il fait partie d'une table dynamique, change la valeur du champ à l'index indiqué (commence à 0)
- runner.getFieldValue(fieldName): Récupère la valeur du champ.
- runner.getFieldValue(fieldName, index): Récupère la valeur du champ à l'index indiqué si le champ est multivalué, ou s'il fait partie d'une table dynamique (commence à 0)

Méthodes utilitaires annexes

- runner.waitForField(fieldName): Le script attend que le champ soit visible dans le document.

Conseil

Cette fonction est utile pour s'assurer qu'un document a terminé d'être chargé.

- runner.randomId(prefix) : Génère une chaîne de caractère aléatoire du type "prefix-ABC-DEF-GHI". Cela permet notamment de retrouver facilement des documents testés si cette valeur est utilisée dans le sujet du document.
- runner.close() : Ferme la fenêtre active du navigateur.
- runner.quit() : Quitte le navigateur
- runner.newTab() : Création d'un nouvel onglet
- runner.switchTab(): Bascule cyclique d'un onglet à l'autre
- runner.switchToFrame(frame): Sélectionne le <frame> HTML courante. La plupart des méthodes de WorkeyScriptRunner se chargent de basculer d'une frame à l'autre si besoin.
- runner.expectAlert(message) : Si une alerte JS est émise, cette fonction permet de la valider et continuer le test.
- runner.pause(message) : Met le script en pause en affichant un message. L'utilisateur doit appuyer sur Entrée pour continuer le test. Cela permet de donner le temps à l'utilisateur d'effectuer des actions non testable (ouverture et vérification d'un fichier externe, par exemple)
- runner.pause(temps_ms) : Marque une pause dans le traitement du script. Utile pour être sûr que certains traitements aient le temps de s'effectuer (JS, etc).
- runner.screenshot(file): Prend une capture d'écran de la fenêtre courante du navigateur et l'enregistre dans un fichier.

Tests JUnit

Les tests se font avec la classe org.junit.Assert

```
// Vérification de la valeur d'un champ
Assert.assertEquals("Valeur de champ correcte",
    "Valeur Correcte", runner.getFieldValue('Champ_a_tester'));
```

- Assert.assertTrue
- Assert.assertFalse
- Assert.assertEquals
- Assert.assertNotNull

Pour vérifier qu'un objet est visible dans le navigateur:

```
Assert.assertTrue("Element #workey-button visible.",
    driver.findElement(By.id('workey-button')).isDisplayed());
```

Expressions XPath

Pour tester une expression XPath dans Chrome, avec la console:

```
document.evaluate("//input[@id='Field_XXX']",
    document, null, XPathResult.ANY_TYPE, null ).iterateNext()
```

Exemple de script

```
import org.junit.Assert;
import org.openqa.selenium.By;
import com.workey.selenium.WorkeyRunnerFactory;

// Création du robot Workey. Le type (v5, v6 ou v7) est déterminé
// automatiquement.
runner = new WorkeyRunnerFactory('validation-selenium.properties').runner;
driver = runner.driver;
logger = runner.logger;

// Connexion avec le login et mot de passe définis dans les propiétés
// user1.login et user1.password
runner.workeyLogin(runner.getCredentials('user1'));

// Création d'un numéro de test aléatoire pour retrouver le document
// facilement lors de la suite des tests
testId = runner.randomId('TestSelenium');
logger.info("Création du document \"${testId}\"." );

// Création du nouveau document
runner.createNewDocument('Document TestSelenium')

// Attente du chargement complet du document
runner.waitForField('Id_Test');
```

```
// Vérification de l'état et du formulaire
runner.assertFormAndState('Formulaire_TestSelenium', '__CREATED');

// Remplissage des champs
runner.setFieldValue('Id_Test', testId);
runner.setFieldValue('String', 'Valeur de mon champ');

// Sélection de l'état suivant et soumission du document
runner.selectNextState('Cree');
runner.clickSubmitButton();

// Ouverture du document dans la vue (peut parfois nécessiter
// de rafraichir la vue)
runner.openDocumentFromView(testId);

// Vérification de la valeur du champ
runner.waitForField('Id_Test');
Assert.assertEquals('Champ String', 'Valeur de mon champ',
    runner.getFieldValue('String'));

// Sélection du dernier état et soumission
runner.selectNextState('Termine');
runner.clickSubmitButton();

// Déconnexion et fermeture du navigateur
runner.workeyLogout();
runner.quit();
```

Module d’Archivage

- Auteur de l’article

Par Briac Date de l’article

- 2021-03-19

Table of Contents

Le portage initial en v.6 de ce module a été réalisé à partir de la v.5.2.38-1 (iso-fonctionnel), avec le strict minimum d'adaptations nécessaires pour son intégration.

Toutefois, certaines fonctionnalités ont été repensées pour adresser la problématique de migration et de continuité des bases d'archives créées en v.5.

Il n'est disponible en v.6 qu'à partir de la **6.11.3**

Principe de fonctionnement

L'objet de ce module est d'extraire de la base socle de Workey les documents qui sont arrivés en fin de vie au regard de leur Workflow respectifs.

Les deux composantes principales du module sont:

1. **L'agent d'archivage**
   Il s'agit d'un Agent Workey spécifique, dont l'action est de créer les bases d'archives, d'identifier les processus à archiver et de réaliser l'extraction des documents éligibles à l'archivage.
   Pour chaque processus archivé, il insère dans la base d'archive désignée les documents extraits et les supprime de la base socle de Workey.
2. **L'application de consultation des archives**
   Il s'agit de l'interface web permettant de rechercher et de visualiser les documents contenus dans les différentes bases d'archives.

Limitations

Warning

Il n'est pas possible de fusionner deux bases archives.

L'archivage d'un processus donné (et de toutes ses versions) ne peut se faire que vers une seule base d'archive désignée.

À éviter

Ne pas désigner **une même base** pour des **processus différents** !

Pré-requis

Le module étant désormais intégré au livrable global du moteur, son accès et sa mise en œuvre sont subordonnés à l'activation via le fichier de licence du serveur Workey.

Information

En cas de mise à niveau du serveur Workey depuis une version antérieure à la v.6, il sera indispensable de mettre à jour le fichier de licence, afin qu'y

soit activée la fonctionnalité d’archivage. A l’occasion de ce portage, le générateur de licence a lui aussi été mis à jour.

Modélisation

Formulaire d’archivage

Depuis la version **5.2.33** de Workey, l’archivage requiert obligatoirement que chaque type document d’un processus dispose d’un Formulaire d’archivage.

Information

La définition de ce Formulaire est primordiale puisque celui-ci sera utilisé pour l’accès au document en vue de son archivage. Seules les données exposées par ce Formulaire seront exportées vers l’archive.

Attention

Corollaire : toute les autres données du document seront définitivement perdues.

Lors de la modélisation d’un Formulaire, il est possible de spécifier si celui-ci sera utilisé comme Formulaire d’écriture et/ou de lecture par défaut, mais aussi comme Formulaire d’archivage.


Il donc possible de définir un Formulaire exclusivement réservé à l’archivage

ou bien un unique Formulaire de Lecture – Écriture – Archivage

Information

Il ne peut y avoir qu’un seul Formulaire d’archivage pour un type de document donné.

Pour les processus déployés dans des versions antérieures à Workey v.6, il sera nécessaire de définir ces formulaires à posteriori, **directement en base**. Pour se faire il faudra identifier, pour chaque type de document et pour chacune des versions du processus, un Formulaire existant (dans la table FORMS) et de valoriser sa colonne IS_FOR_ARCHIVE à 1 ou TRUE (selon le SGBD).

Exemple: soit une base socle ne contenant absolument aucun formulaire, alors désigner les formulaires par défaut (à la fois en Lecture et en Écriture) comme étant les formulaires d’archivage.

```
-- Exemple pour mySQL
UPDATE FORMS SET IS_FOR_ARCHIVE=1 WHERE IS_READ_DEFAULT=1 AND
IS_WRITE_DEFAULT=1;
```

Limitations de la modélisation du formulaire d’archivage

En 6.11.3, certains composants de formulaire ne sont pas supportés lors de la visualisation des données archivées:

- Script JS
- Dossier MG et Documents MG

Certaines caractéristiques ne seront pas prises en compte:

- L’option “Vignette” des champs pièce-jointe
- Le nombre de colonne d’une table

Mise en œuvre

La mise en œuvre du module d’archivage consiste en:

- la désignation des processus à archiver
- la définition d’un agent Workey spécifique

Processus à archiver

Depuis la console d’administration, dans la section “Processus”, en cliquant sur un “Processus déployé”, le détail de celui-ci s’affiche et notamment sa politique d’archivage.

Deux informations sont nécessaires pour que les documents d’un processus soient archivés:

- le nom de la base d’archive cible
- le délai d’archivage

Note: toute modification apportée à ces paramètres doit être validée en cliquant sur le bouton «Modifier l’archivage» pour qu’elle soit effective.

Warning

Il peut arriver qu’une ou plusieurs versions inactives du processus soient susceptibles d’avoir des valeurs de paramétrage différentes de celles de la version active; l’interface le signalera et invitera l’utilisateur à revalider le paramétrage actuel – qui s’appliquera alors à l’ensemble des versions du processus – en cliquant sur « Modifier l’archivage ».

Délai d’archivage

Un processus sera éligible à l’archivage si son délai d’archivage est **égal ou supérieur à 0**.

Ce délai est exprimé en **nombre de jour**.

Un **délai négatif** désactive l’archivage des documents de ce processus.

Lors du premier déploiement d’un processus, le délai d’atchivage est par

défaut fixé à 365 jours. Cette valeur peut toutefois être modifiée à postériori par l'utilisateur. En cas de déploiement d'une nouvelle version du processus existant, alors le délai d'archivage déjà paramétré sera conservé.

Nom de la base d'archive

Les documents du processus archivé seront stockés dans la base d'archive désignée.

Lors du premier déploiement d'un processus, le nom de la base d'archive cible est déterminé depuis le nom interne du processus; les caractères d'espacement y sont substitués par des _ (*underscore*). Ce nom peut toutefois être modifié par l'utilisateur. En cas de déploiement d'une nouvelle version du processus existant, alors le nom de la base d'archive déjà paramétré sera conservé.

Ce nom sera utilisé pour créer un sous-répertoire dans le répertoire global des archives.

Si la base d'archive telle que nommée, n'existe pas, sa création initiale sera assurée par l'agent d'archivage dès qu'un document du processus est archivé.

Attention:

- Le nom de la base d'archive ne doit pas contenir de caractère non autorisé pour la création d'un répertoire sur le système de fichier.
- Le nom ne peut être laissé vide ou constitué exclusivement de caractères d'espacement.
- Tout changement de nom de la base d'archive entraînera la création d'une nouvelle base; à moins de renommer manuellement le répertoire de la base existante, avant le prochain déclenchement de l'agent d'archivage.
- Le changement de base d'archive cible, entre deux déclenchements de l'agent d'archivage, impactera la continuité des archives car les documents nouvellement archivés seront alors stockés dans la nouvelle base. Ceci peut s'avérer très problématique du fait qu'**il n'est pas possible de fusionner/consolider des bases d'archives** entre elles.
- Une base d'archive n'est pas *autonome*; en ce sens que la consultation des documents archivés qu'elle contient s'appuiera sur les droits d'accès propres aux acteurs authentifiés dans Workey.
- Pour la configuration du répertoire global des archives, se reporter à la documentation de la propriété de configuration du serveur com.clog.workey.directory.archives

Agent d'archivage

L'agent d'archivage est un type d'agent Workey spécifique. Lorsque la fonctionnalité d'archivage est activée au niveau de la licence, cet agent sera proposé dans la liste déroulante des types d'agent qui peuvent être créés.

Création de l'agent

Depuis la liste déroulante des types d'agent disponibles, sélectionner

Archiveur et cliquer sur «*Créer nouvel Agent*».

De par son architecture, le module d'archivage repose sur l'action d'**un unique agent** spécifique. Il ne doit y avoir qu'un seul agent d'archivage qui scrute et traite l'ensemble des processus à archiver.

Information

Si un agent d'archivage a déjà été défini, mais que la fonctionnalité d'archivage est désactivée au niveau de la license, alors l'agent se déclenchera au regard de sa programmation mais sera sans effet et s'interrompra aussitôt.

Attention

Ne pas définir plusieurs agents d'archivage car il n'y aura pas de coordination entre les instances de ces différents agents et leurs actions respectives se contrarieraient lors du traitement des documents, aboutissant à une corruption des bases d'archive.

Ne pas définir un agent personnalisé en spécifiant la même classe java que celle de l'agent d'archivage.

Paramétrage de l'agent

L'écran de paramétrage initial de l'agent est le suivant.

Journalisation

L'activation de la **journalisation** de l'agent permettra d'exposer les traitements effectués lors de ses instanciations successives.

Instanciation multiple

En cohérence avec le pré-requis d'unicité de l'agent, les **instances multiples** ne sont pas autorisées.

Paramètre(s) de la tâche

Le paramètre "Nom du répertoire d'archivage" indique, à titre informatif, les nom et chemin du répertoire d'archive global tels qu'ils sont configurés au niveau du serveur Workey. Ce paramètre ne peut pas être modifié au niveau de la configuration de l'agent.

Les paramètres suivants sont optionnels:

- archive.timeout permet de spécifier une durée maximale de traitement de l'agent à chacune de ses instanciations. Ce *timemout* est exprimé en

minutes. Avant d’archiver un document éligible, l’agent s’assure que le temps imparti n’est pas dépassé. Si tel est le cas, alors il interrompra son traitement et abandonnera l’archivage du document et éventuellement de l’ensemble des documents de sa famille.
- archive.lang permet de spécifier une locale particulière lors de l’ouverture des documents à archiver. Bien que le module d’archivage supporte la localisation des processus, ce paramétrage peut avoir une incidence dans des cas très spécifiques de modélisation (ex: connecteur de champ retournant des valeurs déjà localisées et qui seront donc définitivement figées lors de l’archivage). En l’absence de définition de ce paramètre, la langue utilisée sera celle configurée par défaut au niveau du serveur Workey (cf. documentation de la propriété com.clog.workey.engine.defaultLanguage de configuration du serveur Workey).

Identité de l’Agent

L’agent d’archivage utilisera une identité *ad hoc* et spécifique, ayant des privilèges similaires au Gestionnaire de Workflow, pour traiter les documents éligibles. Il n’y a donc aucun paramétrage à effectuer.

Une fois l’agent créé, la page de détail de l’agent mentionnera le détail de l’identité ad hoc.

Information de Déclenchement et Calendrier d’Exclusion

Leurs paramétrages ne présentent aucune spécificité et s’inscrivent dans la même logique que pour les autres Agents Workey.

Structure d’une base d’archive

Les bases d’archives ont une structure commune.

Exemple: d’une base d’archive nommée “ANOMALIES ET RECLAMATION”

Répertoire à ventilation arborescente

Les répertoires attachment/ et documents/ présentent une structure particulière dont la gestion est assurée par le module d’archivage.

Le contenu effectif de ces répertoires sera toujours situé 2 sous-niveaux inférieurs. Ces niveaux intermédiaires sont constitués par des répertoires numérotés/nommés de 00 à FF (en hexadécimal). L’emplacement (chemin) d’un fichier donné sera déterminé au regard d’un algorithme de hachage à partir de son nom.

L’arborescence de répertoire sera créée automatiquement au fil de l’eau selon les besoins de l’agent d’archivage.

Cette stratégie de stockage a été motivée par la volumétrie potentiellement importante des archives.

Répertoire attachments/

C’est sous cette arborescence que seront stockées les pièces jointes des documents archivés.

Au moment de l’archivage, les fichiers correspondants aux pièces jointes sont recopiés vers de nouveaux fichiers avec la nomenclature spécifique suivante:

Concaténation:

- de l’UUID interne de la pièce jointe
- d’un caractère_ (*underscore*) de séparation
- du nom (avec son extension) du fichier d’origine **normalisé** de la pièce jointe

La normalisation du nom du fichier consiste en la «*désaccentuation* » (principe de ramener chaque caractère accentué à sa forme basique) et en la substitution des caractères d’espacement par des _ (*underscore*).

Exemple:
\attachments\b7\50\297e6ed476f1d4f00176f1f17dcb0000_Attestation_validee.pdf

Répertoire documents/

C’est sous cette arborescence que seront stockés les documents archivés avec leurs données.

Au moment de l’archivage, le document est sérialisé au format XML, via l’API Workey, vers un fichier avec la nomenclature suivante :

Concaténation:

- du terme document
- d’un caractère _ (*underscore*) de séparation
- de l’identifiant numérique interne du document
- de l’extension.xml

Exemple:
\documents\0d\0c\document_3773.xml

Répertoire index/

Ce répertoire contient les fichiers de la base *Lucene* utilisée pour indexer le contenu des documents présents dans l’archive, **ainsi que les droits d’accès à ces documents**.

L’indexation est assurée par la même brique fonctionnelle que celle du moteur Workey. Le détail des clefs indexées n’est pas couvert par la présente documentation.

Attention:

- Contrairement à la base *Lucene* adossée à la base socle de Workey, le contenu de celle-ci ne peut être recréé à postériori (pas de réindexation possible). En effet, elle est la seule à détenir les informations de droits d'accès aux document archivés !
- En cas de mise-à-niveau du serveur workey depuis une version antérieure à la v.6, il sera indispensable de migrer la base *Lucene* de chacune des bases d'archive précédemment créées. En effet, les bases d'archives créées sous Workey v.5 utilisaient *Lucene* **2.9.4**. Le module d'archivage sous Workey v.6 utilise *Lucene* **4.8.1**. La migration de structure de ces bases doit être effectuée en deux temps, à l'aide de la classes java org.apache.lucene.index.IndexUpgrader, prévue à cet effet par les concepteurs de *Lucene*:
  - migrer la structure des bases *Lucene* de 2.9.4 → 3.2.0 (nécessite de disposer de la librairie lucene-core-3.2.0.jar dans le classpath java)
  - puis migrer la structure des bases *Lucene* de 3.2.0 → 4.8.1 (nécessite de disposer de la librairie lucene-core-4.8.1 dans le classpath java. Toutefois, celle-ci est déjà indirectement embarquée dans le fichier workey.war du livrable de Workey, sous WEB-INF/lib).

Exemple: des lignes de commande (DOS), pour migrer la base *Lucene* d'une base d'archive. Il est nécessaire de spécifier le chemin vers les fichiers de la base d'index

```
> java -cp lucene-core-3.2.0.jar org.apache.lucene.index.IndexUpgrader -verbose "ANOMALIES ET RECLAMATIONS\index"

> java -cp lucene-core-4.8.1.jar org.apache.lucene.index.IndexUpgrader -verbose "ANOMALIES ET RECLAMATIONS\index"
```

Répertoire locales/

Ce répertoire contient les **ressources de localisation** du processus archivé.

Ces ressources seront utilisées pour la visualisation des documents archivés lors de leur consultation par un utilisateur, au regard de sa préférence de langue.

Au moment de l'archivage, l'agent exportera les ressources de localisation pour chacune des langues supportées/configurées (cf. documentation de la propriété com.clog.workey.engine.supportedLanguages de configuration du serveur Workey), respectivement vers autant de fichiers selon la nomenclature suivante:

Concaténation:

- du terme resources
- d'un caractère _ (*underscore*) de séparation
- de l'**identifiant numérique interne du type de processus**

- d’un caractère _ (*underscore*) de séparation
- du code de la locale
- de l’extension .xml

Exemple:
\locales\resources_16_en.xml
\locales\resources_16_fr.xml
\locales\resources_25_en.xml
\locales\resources_25_fr.xml

Information

Si la liste de langues supportées venait à s’étoffer alors que des documents sont déjà archivés, alors la nouvelle localisation des ressources ne pourra être prise en compte qu’avec les documents d’une nouvelle version du processus.

Répertoire templates/

Ce répertoire contient les modèles (templates) *Velocity* utilisés pour restituer/visualiser les données des documents archivés. Ceux-ci sont désormais ventilés selon qu’ils sont personnalisés ou par défaut.

- custom/, répertoire contenant les templates personnalisés par l’utilisateur.
- default/, répertoire cible de la génération à la volée, par l’agent d’archivage, des templates (version-specific).

Chaque template est propre à un type de document. Toutefois, les templates peuvent être distingué en deux catégories: les **templates génériques** et les **templates spécifiques à une version**. Ils se différencient par la nomenclature des noms de fichier.

Un template générique a pour nom de fichier la concaténation:

- du nom interne du type de document
- de l’extension .vml

Un template spécifique à une version a pour nom la concaténation:

- du nom interne du type de document
- des caractères -v (trait d’union et la lettre v minuscule)
- du **numéro de version du type de processus**
- de l’extension .vml

Attention:

- En cas de mise-à-niveau du serveur workey depuis une version antérieure à la v.6:
  - l’agent d’archivage créera les deux répertoires custom/ et default/, s’ils n’existent pas déjà, lorsqu’un nouveau document

sera archivé à destination de la base d'archive en question.
    - il sera nécessaire de déplacer les templates existant dans le répertoire correspondant en fonction des règles de précédence qui s'appliqueront (cf. sélection des templates).

Fichier metadata.js

{ "process":"Test_Formulaire_Archivage", "delay":0,
"documentTypes":[{"name":"Doc_Test_archive_form","label":"Doc Test archive form"}], "name":"Test_Formulaire_Archivage", "startTime":1614010213303,
"endTime":1614010213603, "fields":[{"name":"Ligne_simple_3","label":"Ligne simple 3"}, {"name":"Les_PJs","label":"Les PJs"},
{"name":"Colonne_2","label":"Colonne 2"},
{"name":"Colonne_1","label":"Colonne 1"},
{"name":"Derniere_colonne","label":"Dernière colonne"}] }

Ce fichier contient la définition d'un objet JSON. Il est initialement généré, et complété au fil du traitement des documents, par l'agent d'archivage. Il contient les métadonnées relatives aux documents présents dans l'archive. Celles-ci seront exploitées par l'application de consultation des archives pour proposer un formulaire de recherche avancée.

- process, le nom interne du type de processus
- name, le nom de la base d'archive
- delay, le délai d'archivage appliqué à lors de la création initiale de l'archive
- documentTypes, un tableau JSON listant les types de document contenus dans l'archive, sous la forme de couples (nom interne, libellé):
    - name, le nom interne du type de document
    - label, le libellé du type de document
- startTime, horodatage de début de la dernière itération d'archivage
- endTime, horodatage de fin de la dernière itération d'archivage
- fields, un tableau JSON listant les champs des document présents dans l'archive, sous la forme de couples (nom interne, libellé):
    - name, le nom interne du champ
    - label, le libellé du champ

Archivage

Rappel: l'agent d'archivage scrute l'ensemble des processus de la base socle. Seuls les processus ayant un délai d'archivage supérieur ou égal à 0 jour sont pris en compte (PROCESS_TYPES.ARCHIVE_DURATION >= 0).

L'agent traite les processus éligibles dans l'ordre alphabétique de leur nom interne.

La structure de la base socle de Workey, avec ses contraintes relationnelles, impose que tous les documents liés par une relation père-fils (dérivation SANS démarrage d'un nouveau processus) doivent être archivés/extraits de la base simultanément (cf. Atomicité de l'archivage).

Pour chaque processus, l'agent commencera par dénombrer les familles de

document (via leur document patriarche) qui répondent aux conditions d'éligibilité à l'archivage. Puis seulement après, il procèdera à l'archivage des documents **famille par famille**.

Les familles de document sont traitées dans l'ordre croissant des identifiants numériques de leur patriarche.

Information

La barre de progression de l'activité de l'agent d'archivage est réinitialisée entre chaque processus traité.

Le message de progression mentionne toujours le processus actuellement traité. Lors de la phase initiale de recherche des documents éligibles à l'archivage, le compteur indique le nombre de documents patriarches analysés; c'est-à-dire le nombre de famille distincte de document.

Dans un second temps, l'agent procède à l'archivage de l'ensemble des documents de ces familles. Le compteur indique alors le nombre de document actuellement traités. Bien entendu, ce nombre pourra être supérieur au nombre de patriarches dans la mesure ou les familles peuvent être constituées de plus d'un document.

Éligibilité des documents

Tous les documents d'une même famille doivent répondre simultanément aux deux contraintes suivantes:

- le document doit **avoir atteint un état final**.
  C'est-à-dire un état pour lequel il n'y a plus d'opération (modélisée) applicable. Le document est **clos**. Concrètement, dans la base socle, le document doit avoir son WORKFLOW.NEXT_STEPS <= 0.
- le document doit avoir atteint son état final **depuis un laps de temps >= au délai d'archivage** défini pour le processus auquel il appartient. Pour cela, l'horodatage WORKFLOW.LAST_ACCESS_TIME du document sera confronté à la date courante.

Information

Il n'est pas possible d'exclure un type de document donné pour un processus archivé. Tous les types de documents seront pris en considération.

Atomicité de l'archivage

L'archivage d'une famille de document est **un traitement atomique**. Si l'archivage de l'un des documents échoue ou est interrompu, alors aucun des documents de la famille ne sera archivé. Le périmètre de ce traitement englobe les actions suivantes.

Vérification d’interruption

Avant d’entamer le traitement d’un document donné, il y a vérification des éventuels sémaphores d’interruption. **Toute interruption levée mettra fin à l’instance en cours de l’agent d’archivage**.

Limitation du nombre de document

A des fins de débogage, il est possible de définir un nombre maximum de document que l’agent d’archivage est autorisé à traiter à chaque déclenchement, via la propriété de configuration du serveur: com.clog.workey.archive.debug.docLimit.

Information

Cette limitation s’applique par processus.

Timeout global

Le paramétrage optionnel de l’agent permet de spécifier un temps de traitement global maximum pour une instanciation de l’agent (cf. paramètre archive.timeout).

Information

Cette limitation s’applique à l’instance de l’agent, quelque soit son avancée dans le traitement des différents processus.

Interruption à la demande de l’utilisateur

Depuis le tableau de bord des Agents dans la console d’administration du serveur Workey, il est possible d’interrompre l’instance en cours d’exécution de l’agent d’archivage en cliquant sur la croix rouge située à droite de sa barre de progression. Cette demande d’interruption est donc scrutée par l’agent avant de commencer à traiter un document.

Éviction des documents avortés

Si le document à traiter n’a jamais progressé dans le flux, alors il n’est pas archivé et sera supprimé. Typiquement, le document est resté dans l’état système _CREATED ou _DERIVED.

Parcours récursif arborescent

L’archivage d’une famille a pour point de départ le document patriarche. En revanche, l’archivage d’un document traite d’abord sa descendance avant d’archiver le document en lui même. Les documents fils sont traités dans l’ordre croissant de leur identifiant numérique. L’opération d’archivage prend donc la forme d’un traitement récursif du fait du parcours arborescent de la hiérarchie père-fils.

Accès au document pour archivage

Pour archiver un document, il est nécessaire de l’accéder en lecture à l’aide

du Formulaire d’archivage défini pour ce type de document.

Le document est accédé avec les **privilèges d’un Gestionnaire de Workflow**.

Warning

L’absence de formulaire d’archivage est fatal dans le traitement du document. L’instance de **l’agent d’archivage sera alors interrompue**.

Génération modèle *Velocity*

Information

Ce mécanisme est une refonte de la fonctionnalité précédemment accessible depuis l’application de consultation des archives.

Le module d’archivage de Workey v.5 nécessitait obligatoirement de disposer des fichiers xml de déploiement. Avec le passage du modélisateur *GraphTalk* aux modélisateurs FLASH puis HTML5, la structure de ces fichiers a évolué à plusieurs reprises, rendant impossible de proposer une mécanique homogène et adaptative de génération des modèles (templates) *Velocity*.

C’est pourquoi une **génération automatique de modèles par défaut** a été mise en place. Cette génération est réalisée, au fil de l’eau, lors de l’archivage des documents. Pour un document donné, et **spécifiquement pour la version du processus à laquelle il appartient**, si aucun template par défaut n’existe déjà dans le répertoire templates/default/, alors un modèle est créé **à partir du Formulaire d’archivage**. vec une nomenclature spécifique.

Concaténation:

- du nom interne du type de document
- des caractères -v (trait d’union et la lettre v minuscule) de séparation
- du **numéro de version du type de processus**
- de l’extension .vml

Attention

La personnalisation des modèles *Velocity* n’est pas traitée dans cette présente documentation. Elle requiert des connaissances techniques spécifiques.

Mise-à-jour du /metadata.js

Lors de l’archivage d’un document, les métadonnées sont complétées. S’ils ne sont pas déjà présents dans le fichier, les noms internes et les libellés des types de document et champs archivés sont ajoutés aux tableaux JSON correspondants: documentTypes et fields.

Export XML du document

Le document ouvert est exporté au format xml vers un fichier placé dans le répertoire documents/ de l’archive. La sérialisation est assurée par l’API

Java de Workey. Un document archivé reprend donc la structure standard avec quelques informations complémentaires.

L'élément <wky:document> est complété:

- des attributs:
  - parent-id, l'identifiant numérique du document père (uniquement si le document présente un document père)
  - archive-date, la date à laquelle le document a été archivée (date normalisée selon le modèle yyyy-MM-dd'T'HH:mm:ss.SSS).
  - hash-dir, le chemin relatif depuis le répertoire documents/ vers le présent fichier xml.
- et le cas échéant, d'un sous élément <children-documents>, contenant lui même autant de sous-élements <child> ayant en attribut l'id de chacun de ses fils directs.

Exemple:
parent-id="968" archive-date="2017-06-01T15:07:28.941" hash-dir="0d/0c"

Indexation

Rappel: l'indexation est assurée par la même brique fonctionnelle que celle du moteur Workey.

En revanche, pour les besoins spécifiques au module d'archivage, sont aussi indexées les informations suivantes:

- la **date d'archivage** du document; clé: Workey_ArchiveDate
- les **droits d'accès** au document archivé:
  - **nominatifs**; désignation des acteurs par leur identifiant numérique (ACTORS.ID); clé: Workey_PRK
  - **publics**; c'est-à-dire via les noms internes des rôles (ROLES.DESIGNER_NAME); clé: Workey_PublicAccessRoleDesignerName
- la **clef de hachage** permettant d'accéder au fichier xml de données du document dans l'arborescence du répertoire documents/; clé: Workey_HashDir
- les **relations père-fils**, avec la liste des identifiants numériques des documents fils; clé: Workey_ChildDocId

Export des pièces jointes

Chacune des pièces jointes du document archivé est recopiée vers le répertoire attachments/ de l'archive, avec renommage (cf. nomenclature des fichiers de ce répertoire).

Information

Si une même pièce jointe est partagée entre plusieurs documents d'une même famille, alors cette recopie aura lieu autant de fois qu'elle est référencée; le fichier existant étant écrasé par la nouvelle recopie.

Export des ressources de localisation

A chaque document archivé, appartenant donc à une version précise d'un type de processus, la présence de ressources de localisation est vérifiée. Si aucune ressource n'est trouvée, alors il a création d'autant de fichiers de ressource qu'il y a de langues supportées dans la configuration du serveur. Ces fichiers sont créés dans le répertoire locales/ avec la nomenclature détaillée ici.

Rollback

L'atomicité implique l'obligation d'un retour arrière (*rollback*) en cas d'échec de l'archivage d'un document ou en cas d'interruption.

Le périmètre de ce *rollback* est défini par les documents de la famille en cours de traitement lors de la survenue de l'exception (erreur, échec, interruption, etc…). Les opérations menées en cas de *rollback* sont les suivantes:

- suppression des fichiers nouvellement créés dans les arborescences documents/ et attachments/. En revanche, il n'y aura pas suppression des répertoires intermédiaires, car ceux-ci sont potentiellement utilisés par les clefs de hachages d'autres documents déjà archivés avec succès.
- la base d'index n'est pas mise à jour avec le résultat de l'indexation des documents de la famille.

Ne sont pas concernés par le *rollback* les éléments découlant directement des métadonnées du processus:

- les modèles *Velocity* par défaut qui auraient été générés.
- les fichiers de ressources de localisation nouvellement exportés,
- les metadonnées des éléments de recherche (le fichier /metadata.js).

Suppression

Lorsque l'exportation des documents d'une famille, vers la base d'archive, est réalisée avec succès, alors ceux-ci peuvent être supprimés de la base socle du serveur Workey. Cette suppression est nécessairement hors-périmètre de l'atomicité et englobe les actions suivantes:

- suppression des tuples correspondant au(x) document(s) dans les différentes tables de la base socle. Ces suppressions sont effectuées à l'aide de requête SQL (*batchées*) ciblant les tuples par leur clef primaire.
- invalidation des caches d'*Hibernate*, pour éviter toute désynchronisation avec la base directement modifiée via JDBC.
- suppression des éventuelles pièces jointes pièces jointes du serveur Workey. Cette suppression repose sur la brique fonctionnelle du moteur gérant les pièces jointes (recours à la fonction de purge).
- désindexation des documents de la base Lucene du serveur Workey.

E-mail aux Gestionnaires de Workflow

En cas d'interruption du traitement d'archivage, et uniquement dans ce cas, un email d'alerte est envoyés aux seuls acteurs (de classe LDAP 'Person') disposant des droits Gestionnaire de Workflow.

Cette notification a pour sujet (localisé au regard de la langue par défaut paramétrée au niveau du serveur Workey):

En français: L'agent d'archivage Workey a rencontré une erreur.
En anglais: The Workey archiving agent has encountered an error.

Toutefois, le corps du mail est en anglais et expose les éléments techniques détaillant la nature de l'interruption, ainsi que l'état d'avancement de l'archivage au moment de sa survenue.

Exemple:

Exemple de mail adressé aux Gestionnaires de Workflow.

Les éléments techniques sont les suivants:

- l'URL à laquelle le serveur est censé être joignable. Il s'agit ni plus ni moins de la valeur de la propriété com.clog.workey.engine.ExternalURL.serverContext de configuration du serveur.
- l'identifiant numérique de l'Agent d'archivage
- la date de déclenchement de l'instance de l'Agent
- la date d'interruption du traitement d'archivage
- la liste des processus déjà traités durant cette exécution. Pour chacun, le nom interne du processus, avec le nombre total de documents traités, ventilés le cas échéant:
  - les documents archivés avec succès (avec le tableau des identifiants numériques des documents)
  - les documents ignorés (cf. éviction des documents avortés)
- Si une exception Java a été jetée, alors celle-ci est explicitée:
  - Failure message : le message d'erreur remonté
  - Root cause : la cause racine, en cas d'encapsulation
  - Full stack trace follows : la pile d'exécution à l'instant de l'exception.

Application de consultation des archives

Accès à l'application

La servlet de l'application est embarquée et définie dans le même fichier workey.war que l'application Web de Workey. Son accès est sécurisé par le même module d'authentification. En conséquence, un utilisateur authentifié dans l'application Web, le sera automatiquement dans l'application de consultation des archives.

Information

L’accès à l’application de consultation des archives est conditionné à l’activation de la fonctionnalité d’archivage au niveau de la licence. Dans le cas contraire, l’utilisateur recevra un code d’erreur HTTP 503 en retour.

Il y a deux moyens pour accéder à cette interface:

- soit directement via l’URL de la page d’accueil: /workey/archives/home
- soit depuis l’application Web de Workey, via le menu utilisateur. A noter que le lien ne sera proposé que si la fonctionnalité d’archivage est activé au niveau de la licence


Lien d’accès vers l’application de consultation des archives, depuis l’application Web de Workey

Information

Du point de vue de Spring Security, un utilisateur authentifié dans Workey se verra attribuer le rôle WK_ARCHIVES_ROLE si l’archivage est activé au niveau de la licence, en plus du rôle WORKEY_ROLE.

Page d’accueil

La page principale consiste en un formulaire de recherche avancée. L’utilisateur est invité à sélectionner la base d’archive qu’il souhaite requêter., depuis la liste déroulante alimentée avec les noms des bases d’archives présentes dans le répertoire d’archive global.

Les liste déroulantes “Type de document” et “Champs” sont peuplées avec les informations contenues dans le fichier /metadata.js de l’Archive sélectionnée.

Information

La liste de “Champs” n’est pas dépendante de la sélection réalisée dans la liste “Type de document”.

Recherche d’un document archivé

En cas de recherche effectuée par un utilisateur ne disposant pas de privilèges de Gestionnaire de Workflow, l’application d’un filtre sur ses droits spécifiques – tant nominatifs, que publics du fait de ses affectations aux Rôles – sera assurée de la même manière que pour une recherche de documents dans l’application Web de Workey.

Le résultat de la recherche est trié par ordre décroissant de pertinence.

Une limitation hardcodée a été fixée à 10000 résultats.

Consultation d’un document archivé

L’utilisateur peut accéder à un document en cliquant sur son sujet depuis la liste des résultats de sa recherche.

Détermination du modèle *Velocity*

Au regard du **type de document** et de la **version du processus** auquel appartient le document accédé, la servlet va sélectionner un modèle *Velocity* présent dans l’arborescence des répertoires templates/.

Parmi les modèles éligibles, l’algorithme sélectionnera un modèle en privilégiant toujours:

- un modèle version-spécifique sur un modèle générique
- un modèle personnalisé sur un modèle par défaut

Il découle de ces deux principes l’ordre préférentiel suivant:

Information

Pour les documents archivés à partir de la v.6, l’agent d’archivage génère automatiquement un modèle version-spécifique par défaut pour chaque type de document traité.

Pour les documents archivés avant la v.6, leurs modèles devront être correctement localisés dans les sous-répertoire de templates/ et répondre à la nomenclature correspondant à leur spécificité (générique ou version-spécifique).

Attention

Un modèle est **indispensable** pour générer le code HTML de la page affichant le document consulté.

Contexte de la fusion *Velocity*

Pour desservir le document consulté, l’application réalise une fusion des données du document et des ressources de localisation à l’aide d’un modèle *Velocity*.

Les données des fichiers xml sont injectées, sous forme de couples clé/objet, dans une Map (le contexte) qui sera passée en paramètre à *Velocity*.

Injection des ressources localisées

Au regard de la langue préférentielle de l’utilisateur (propriété language de l’objet UserAuth issu de l’authentification), le contenu du fichier de ressources correspondant est ajouté à la Map du contexte *Velocity*.

Chacun des éléments de localisation du fichier est ajouté à une nouvelle Map avec une clef constituée par concaténation:

- du type de la ressource
- du caractère séparateur | (*pipe*)
- du nom interne de la ressource

Cette objet Map, regroupant tous les éléments de localisation, est ajouté au du contexte *Velocity* avec la clé langRes.

Exemple:

```
{   ...
    langRes: {
        document-field|demandeur: "Demandeur",
        document-field|modele: "Référence du modèle",
        ...
        operation|Creer : "Créer"
        operation|Finir : "Finir"
        ...
        state|cree: "créé"
        ...
        role|responsable: "Responsable validation"
        ...
    }
    ...
}
```

Une exception s’applique pour les éléments des vues embarquées; notament pour les clefs des noms localisés des colonnes. Celles-ci sont constituées par concaténation des éléments suivants:

- du type de la ressource, en l’occurrence view-column
- du caractère séparateur | (*pipe*)
- du nom interne de la vue embarquée
- du caractère séparateur | (*pipe*)
- du nom interne de la colonne

Exemple:

```
{   ...
    view-column|vue_docs_fils_action|N__Traitement_ou_Action : "N° Traitement
ou Action"
    view-column|vue_docs_fils_action|COL_1: "Libellé",
    view-column|vue_docs_fils_action|COL_SUBJECT: "Objet de l'action",
    view-column|vue_docs_fils_action|COL_ID_DOC: "Référence de l'action",
    ...
}
```

Injection des données du document

Pour chaque champ du document archivé, un objet représentant les valeurs du champ sera associé (dans la Map du contexte) au nom interne du champ. Cet

objet sera la liste ordonnée (ArrayList, potentiellement vide) des valeurs du champ.

```
{
   modele: [],
   couleurs_selectionnees: ["bleu","vert"],
   demandeur: ["Jean"],
   date_demande: ["2022-01-01T00:00:00.000"],
   ...
}
```

Les valeurs seront généralement des chaînes de caractère (String). Toutefois si le champ est de type pièce jointe, alors les valeurs de la liste seront elles-même des Map détaillant les caractéristiques de chacun des fichiers.

```
{  ...
   demandeur: ["Jean"],
   date_demande: ["2022-01-01T00:00:00.000"],
   justificatifs: [
     {  name: "sa25-général.pdf"
        type: "application/pdf"
        uuid: "4C0A66D8AC168804016E2194636DE7C4"
        href: "attachments/1111/4C0A66D8AC168804016E2194636DE7C4_sa25-
general.pdf"
     },
     {  name: "test.pdf"
        type: "application/pdf"
        uuid: "A7BFCED2AC168804016E2194D8FA6B99"
        href: "attachments/1111/A7BFCED2AC168804016E2194D8FA6B99_test.pdf"
     },
   ...
}
```

- name, le nom original du fichier (avec son extension)
- type, le type MIME du fichier
- uuid, l'identifiant unique de la pièce jointe
- href, l'URL relative qui devra être utilisée pour télécharger le fichier de la pièce jointe. Celle-ci est constitué de la concaténation des éléments suivants:
  - la chaîne attachments/
  - l'identifiant numérique du document suivi d'un /
  - l'uuid du document
  - un caractère _ (*underscore*) séparateur
  - le nom normalisé du fichier (cf. normalisation du nom du fichier original)

Information

Les valeurs des champs de type date sont sérialisées par l'API Workey au format standardisé ISO 8601: yyyy-MM-dd'T'HH:mm:ss.SSS

Injection des données des vues embarquées

Le contenu des vues embarquées est archivée sous forme matricielle (une liste de liste de valeurs textuelles). Chaque vue sera associée, par le biais de son nom interne, à la liste ordonnées (ArrayList, potentiellement vide) des lignes qui la constitue; chaque ligne étant elle même un objet (une Map) dans lequel les valeurs textuelles sont associée au nom interne de leur colonne.

Exemple:

```
{   vue_docs_fils_action: [
          {   COL_1: "",
              COL_SUBJECT: "Fils n°1253545 pour action",
              COL_ID_DOC: "123545",
              Workey_Id: "123545"
              Workey_Subject: "Fils n°1253545 pour action"
          },
          {   COL_1: "",
              COL_SUBJECT: "Fils n°127426 pour action",
              COL_ID_DOC: "127426",
              Workey_Id: "127426"
              Workey_Subject: "Fils n°127426 pour action"
          },
          ...
       ],
    ...
}
```

A noter que chaque ligne d'une vue correspond à un document Workey; c'est pourquoi l'objet représentant une ligne contient, en plus des valeurs des colonnes, deux informations complémentaires:

- Workey_Id, l'identifiant interne du document Workey pour cette ligne
- Workey_Subject, le sujet du document Workey pour cette ligne

Injection de l'historique du document

Le détail de l'historique du document est ajouté à la Map du contexte sous la forme d'une liste ordonnée (ArrayList, théoriquement jamais vide) associé à la clé history. Chaque valeur de cette liste est un objet (une Map) contenant le détail d'un entrée d'historique.

Exemple:

```
{   ...
    history: [
          {   isoDate: "2020-12-07T10:53:38.030",
              date: @java.Util.Date,
              actor: "Jean",
              operation: "Creer",
              state: "Cree"
          },
```

```
        {   isoDate: "2020-12-07T10:54:24.530",
            date: @java.Util.Date,
            actor: "Jean",
            operation: "Finir",
            state: "Termine"
        },
        ...
    ]
}
```

Les propriétés (couples clé/valeur) de ces entrées d'historique sont les suivantes. Sauf exception, les valeurs sont des chaînes de caractère (String)

- isoDate, la date de réalisation de l'opération, normalisé ISO 8601
- date, une instance de java.util.Date représentant la même date que isoDate
- actor, le nom de l'acteur ayant réalisé l'opération
- operation, le nom interne de l'opération réalisé
- state, le nom interne de l'état atteint à l'issue de l'opération

Injection des informations générales

En complément, le contexte est peuplé d'un ensemble d'informations communes aux documents Workey.

- Workey_Id, l'identifiant numérique du document
- Workey_Subject, le sujet du document
- Workey_Process, le nom interne du type de processus auquel appartient ce document
- Workey_DocType, le nom interne du type de document dont relève ce document
- Workey_State, le nom interne de l'état final du document
- Workey_ParentId, l'identifiant numérique de l'éventuel document père (peut être null)
- Workey_ArchiveDate, la date d'archivage du document (formatée ISO 8601)
- Workey_ChildrenDocuments, uniquement si le document présente des documents fils. La valeur associée est alors la liste (ArrayList) des identifiants numériques des documents fils (String)

Exemple:

```
{   ...
    Workey_Id: "123544",
    Workey_Subject: "Doc N°123544",
    Workey_Process: "Processus_de_test",
    Workey_DocType: "Mon_type_de_document"
    Workey_State: "Termine",
    Workey_ParentId: null,
    Workey_ArchiveDate: "2021-02-22T17:10:13.399
    Workey_ChildrenDocuments: ["123545","127426"],
    ...
}
```

Limitations de la visualisation des documents archivés

Aux différentes limitations (du module d'archivage ou de la modélisation) s'ajoutent les limitations suivantes, spécifiques à la visualisation des documents archivés.

Les listes de valeur des domaines associés aux champs ne sont pas disponibles une fois le document archivé. Par conséquent, toutes les apparences liées à un choix (multiple ou non), telles que les case-à-cocher, les ensembles de boutons radio, les listes déroulante, etc. ne seront pas prises en charge. Seule la ou les valeurs sélectionnées seront affichées.

Certaines mises en forme définies dans le formulaire d'archivage ne sont pas conservées. C'est notamment le cas des champs disposés dans des tables statiques; leur ventilation par colonne (ou par césure en v.5) n'est pas reprise dans les modèles par défaut générés par l'agent d'archivage.

À éviter

Les données exposées par le formulaire d'archivage seront visibles par l'ensemble est utilisateurs ayant accès au document archivé !

Si du code javascript était utilisé pour contrôler la visibilité de certains éléments du formulaire (masquage de champs ou de sections)**, alors celui-ci sera caduque car inopérant lors de la consultation des documents archivés !**

Modèles *Velocity*

La création ou la modification de modèle *Velocity* ne sera pas traité dans le cadre de cette documentation. Pour de plus amples informations: https://velocity.apache.org/

Macros spécifiques

Afin de faciliter l'exploitation et la restitution des données archivées injectées dans le contexte *Velocity*, un ensemble de macros ont été définies. Les modèles générés automatiquement par l'agent d'archivage s'appuient sur ces macros. Pour plus de détail sur l'implémentation de ces macros, se référer au fichier VM_archive.vm situé à l'intérieur de l'archive web workey.war sous le chemin: /WEB-INF/classes/com/workey/archiver/templates/

Macro (paramètres…)	Description
#label($langKey, $defaultLabel)	Retourne l'élément de localisation correspondant à la clé $langKey passée en premier paramètre. Si aucune localisation n'est associée à cette clé dans la map langRes du contexte, alors retourne le libellé $defaultLabel passé en second paramètre.

Macro (paramètres…)	Description
#fieldValue($field, $index)	Retourne la valeur, à la position $index, du champ désigné par $field. La première valeur à pour position d'index 0 (zéro). Ne retourne rien, si le champ ne contient aucune valeur. Retourne la première valeur si le paramètre $index est omis. Retourne - si la position d'index demandée excède le nombre de valeur du champ.
#fieldMultiValue($field)	Retourne l'ensemble des valeurs du champ désigné par $field, sous la forme d'une UnsortedList HTML (élément). Chaque valeur étant un item de liste (élement). Ne retourne rien, si le champ ne contient aucune valeur.
#dynamicTable($columnNames)	Retourne les lignes d'un tableau dynamique. Le paramètre obligatoire, $columnNames, est un tableau contenant les noms des champs constituant le tableau dynamique. Pour chaque ligne du tableau, un élément <tr> est créé, avec autant de cellules (<td>) qu'il y a d'éléments dans $columnNames. Il est impératif que tous les champs du tableau aient le même nombre de valeur. Sachant que le nombre de valeur du premier des champs détermine le nombre de ligne total du tableau. **Note**: cette macro fait elle même appel à la macro #fieldValue. Son usage doit se faire au sein d'un élément <table> ou <tbody>.
#linkAttachment($field, $index)	Retourne un lien relatif, sous la forme d'un élément <a>, d'accès à la pièce jointe à la position $index du champ désigné par $field. Le contenu de l'élément correspond au nom original du fichier. L'attribut href est renseigné avec la valeur correspondant à la clé href dans le contexte *Velocity*. Contrairement à la macro #fieldValue, l'appelant doit s'assurer que la position d'index n'excède pas le nombre de valeur du champ.
#linkAttachmentMV($field)	Retourne l'ensemble des liens d'accès aux pièces jointes du champ désigné par $field, sous la forme d'une UnsortedList HTML (élément). Chaque élément <a> étant encapsulé dans un item de liste (élement). Ne retourne rien, si le champ ne contient aucune valeur. **Note**: cette macro fait elle même appel à la macro #linkAttachment.
#viewContent($view, $columnNames)	Retourne les lignes de la vue embarquée désignée par $view. Le paramètre obligatoire, $columnNames, est un tableau contenant les noms internes des colonnes constituant la vue. Pour chaque ligne de la vue, un élément <tr> est créé, avec autant de cellules (<td>) qu'il y a d'éléments dans $columnNames. **Note**: son usage doit se faire au sein d'un élément <table> ou <tbody>.
#isEmpty($field)	Retourne false si le champ $field existe et qu'il contient au moins une valeur et que la première valeur est non vide. Retourne true dans le cas contraire.

Macro (paramètres…)	Description
#formatDate($field, $index)	Retourne la date, à la position $index, du champ désigné par $field. La première date à pour position d'index 0 (zéro). La date est formatée de sorte à ne retenir que la partie calendaire (pas d'information d'heure). Le formatage est localisé par rapport à la langue préférentielle de l'utilisateur. Le pattern utilisé correspond à celui défini par java.text.DateFormat.LONG. **Note**: macro destinée à être utilisée avec les valeurs des champs de type date (qui ont été sérialisées/normalisées ISO 8601 lors de l'archivage).
#formatDateTime($string)	Retourne la date et l'heure reformatés, à partir d'une date sérialisée/normalisée ISO 8601 Le formatage est localisé par rapport à la langue préférentielle de l'utilisateur. Le pattern utilisé correspond à celui défini par java.text.DateFormat.MEDIUM à la fois pour la date et pour l'heure. **Note**: macro destiné à être utilisé avec les valeurs des champs de type date (qui ont été sérialisées/normalisées ISO 8601 lors de l'archivage).
#documentHistory	Retourne les lignes du tableau d'historique du document archivé. Chaque entrée d'historique est traduite en élément <tr>, avec des éléments <td> pour: .la date et l'heure (#formatDateTime), .l'acteur ayant réalisé l'opération, .le nom localisé de l'opération, .le nom localisé de l'état atteint. Se charge d'ajouter, en dernière entrée, l'opération d'archivage. **Note**: son usage doit se faire au sein d'un élément <table> ou <tbody>.
#startHTML	Insère en tant qu'entête le template interne /WEB-INF/classes/com/workey/archiver/templates/header.vml situé à l'intérieur de l'archive web workey.war
#endHTML	Insère en tant que bas de page le template interne: /WEB-INF/classes/com/workey/archiver/templates/footer.vml situé à l'intérieur de l'archive web workey.war

Debug

Packages et classes java pouvant être tracés dans les logs:

- pour l'agent d'archivage (celui-ci fait appel aux fonctionnalités noyau d'archivage): com.clog.workey.agents.tasks.ProcessArchiver
- pour les fonctionnalités noyau d'archivage: com.clog.workey.archive
- pour l'application de consultation des archives: com.workey.archiver
- pour la fusion *Velocity*: com.workey.archiver.Velocity

Signature électronique

- Auteur de l'article

Par Briac Date de l'article

- 2021-03-16

Module de signature PDF

Installation

Le module de signature électronique est disponible à partir de la version **6.11.4** de Workey.

Ce composant s'appuie sur les bibliothèques de cryptographie Bouncy Castle. Celles-ci doivent être installées dans le JRE ($JAVA_HOME/jre/lib/ext/) et déclarées dans le fichier $JAVA_HOME/jre/lib/security/java.security en ajoutant une ligne :

```
security.provider.2=org.bouncycastle.jce.provider.BouncyCastleProvider
```

Libersign et extensions

Afin de pouvoir effectuer la signature qualifiée avec le certificat de l'utilisateur, il est impératif d'installer au préalable l'application Libersign, ainsi que l'extension appropriée au navigateur utilisé.

Application Libersign

- Libersign

Extensions navigateurs

- Extension Chrome/Edge
- Extension Firefox

Information

Sur Windows 10, l'application Libsergin est installée dans le répertoire "%LOCALAPPDATA%\Libersign". Il contient un exécutable JRE ainsi que l'applet dans "%LOCALAPPDATA%\LiberSign\applet\SplittedSignatureApplet.jar".

Signature simple – cachet serveur

Connecteur com.clog.workey.connectors.parapheur.SignaturePDF

Configuration du connecteur

Ce connecteur nécessite qu'un fichier keystore contenant la clé à utiliser soit présent sur le serveur. Plusieurs propriétés doivent être configurées:

- com.clog.workey.connectors.parapheur.SignaturePDF.KEYSTORE_FILE : Emplacement du fichier keystore à utiliser (**obligatoire**)
- com.clog.workey.connectors.parapheur.SignaturePDF.ALIAS : Alias présent ans le keystore à utiliser pour la signature (**obligatoire**)
- com.clog.workey.connectors.parapheur.SignaturePDF.PASSWORD : Mot de passe de l'alias (**obligatoire**)
- com.clog.workey.connectors.parapheur.SignaturePDF.HASH_ALGORITHM : Algorithme de hash à utiliser ("SHA-1" par défaut)
- com.clog.workey.connectors.parapheur.SignaturePDF.KEYSTORE_TYPE : Type du keystore ("PKCS12" par défaut)
- com.clog.workey.connectors.parapheur.SignaturePDF.KEYSTORE_PROVIDER : Type du fournisseur de keystore

par exemple:

```
com.clog.workey.connectors.parapheur.SignaturePDF.KEYSTORE_FILE=/chemin/vers/un/fichier/test_store
com.clog.workey.connectors.parapheur.SignaturePDF.ALIAS=Workey Parapheur
com.clog.workey.connectors.parapheur.SignaturePDF.PASSWORD=XXXXXXXXXXXXX
com.clog.workey.connectors.parapheur.SignaturePDF.HASH_ALGORITHM=SHA1
com.clog.workey.connectors.parapheur.SignaturePDF.KEYSTORE_TYPE=pkcs12
#com.clog.workey.connectors.parapheur.SignaturePDF.KEYSTORE_PROVIDER=
com.clog.workey.connectors.parapheur.SignaturePDF.image=/chemin/vers/une/image.png
```

En entrée

1. Le champ PJ contenant les PDF à signer
2. Un objet JSON précisant les options de signature (toutes les propriétés sont optionnelles):

Propriété	Description
name	Nom du signataire.
location	Endroit (ou nom du serveur) où s'effectue la signature.
reason	Objet de la signature.
tsa	URL vers un serveur TSA.
index	Si le PDF doit être signé par plusieurs personnes et que les signatures sont visibles, ce champ doit indiquer l'index de la signature en cours (démarre à 0).
invisible	Booléen indiquant si la signature doit être invisible.
box	Objet JSON décrivant le rectangle de signature visible

Propriété	Description
signaturePage	Entier indiquant le numéro de page de la signature (-1 pour la dernière page, 0 pour créer une nouvelle page à la fin du document).
templateString	Modèle du libellé de signature, les balises “{{DN}}” et “{{DATE}}” sont remplacés par l’identité du signataire et la date de signature, respectivement
dateFormat	Format de la date à utiliser pour le modèle ci-dessus

Propriétés JSON de configuration de signature électronique

Propriété	Description
columns	Nombre de colonnes de tampons de signature par page (par défaut: 3).
startX	Coordonnée horizontale de départ du premier rectangle (par défaut: 50).
startY	Coordonnée verticale de départ du premier rectangle (par défaut: 50).
width	Largeur du rectangle de signature (par défaut: 150).
widthMargin	Marge horizontale entre deux rectangles de signature (par défaut: 20).
height	Hauteur du rectangle de signature (par défaut: 50).
heightMargin	Marge verticale entre deux rectangles de signature (par défaut: 20).
backgroundColor	Code hexadécimal de la couleur de fond du rectangle de signature (par défaut: “#fafafa“).

Propriétés JSON du rectangle de signature

Utilisation des propriétés du rectangle de signature

Image de fond

Il est également possible de configurer une image à afficher comme fond du rectangle. Pour cela, il faut renseigner la propriété système suivante avec le chemin vers le fichier image à inclure:
com.clog.workey.connectors.parapheur.SignaturePDF.image

Voici un exemple de champ de configuration de signature électronique :

```
{
  "invisible": false,
  "box": {
    "columns": 2,
    "width": 200,
    "height": 75,
  },
  "signaturePage": -1,
  "templateString": "Signed By: {{DN}}\nOn: {{DATE}}",
  "dateFormat": "yyyy-MM-dd HH:mm:ss z"
```

```
}
```

En sortie

1. *(aucun)* Le contenu des pièces-jointes signées est mis à jour

Signature qualifiée

La signature qualifiée dans le module de parapheur Workey se déroule en deux étapes : la préparation et la signature proprement dite. Cette séparation permet de signer les documents sans avoir à les transférer en intégralité sur le poste client.

Étape 1 – Préparation des fichiers PDF à signer

Le connecteur com.clog.workey.connectors.parapheur.PreparePDFSignature permet de préparer des fichiers PDF à être signés par signature qualifiée.

Attention

Il est important qu'une fois ce connecteur exécuté, les pièces-jointes concernées ne doivent pas être modifiées. Cela rendrait la future signature invalide.

En entrée

- Champ de type pièce-jointe. Peut être un champ multivalué.
- (*optionnel*) Configuration JSON de la signature, du même type que la signature qualifiée. Par exemple :

```
{
  "name": "Jean Dupont",
  "reason": "Signature contrat de travail",
  "location": "Bayonne, France",
  "tsa": "http://tsa-provider/rfc3161"
}
```

En sortie

- JSON contenant la liste des UUID et leur digest respectif. Par exemple :

```
[
  { "uuid": "uuid_1", "digest": "digest_1" },
  { "uuid": "uuid_2", "digest": "digest_2" },
  ...
]
```

Script formulaire

Dans la version actuelle du module de parapheur, il est nécessaire de configurer un fichier JavaScript spécifique à chaque formulaire devant recevoir une signature qualifiée.

Ce script doit être placé à l'endroit suivant: $TOMCAT_HOME/workey/custom-js/{NOM_DU_PROCESSUS}/{NOM_DU_DOCUMENT}/{NOM_DU_FORMULAIRE}/form.js

Dans les premières lignes, il faut modifier les deux variables suivantes pour qu'elle correspondent à

```
$Workey$digestField = 'JSON_Signature';
$Workey$libersignField = 'Libersign';
```

Étape 2 – Signature des fichiers PDF

Le connecteur com.clog.workey.connectors.parapheur.AddSignature permet d'insérer la signature effectuée par Libersign dans le document PDF préalablement préparé.

En entrée

1. Champ JSON comportant les données de signature. C'est celui qui est en sortie de l'étape 1.
2. (*optionnel*) Configuration JSON de la signature, du même type que la signature qualifiée.

En sortie

- (*Aucun*) Le contenu des pièces-jointes signées est mis à jour.

Agent d'extraction Workey

- Auteur de l'article

Par Briac Date de l'article

- 2021-03-16

Table of contents

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.Lire la suite "Agent d'extraction Workey"

- Étiquettes
- agent

Changelog version 6.11.4

- Auteur de l'article

Par Briac Date de l'article

- 2021-03-16

Listes des nouvelles fonctionnalités, modifications ou corrections de la prochaine version. Liens vers le changelog v6.

Avant de tagger, bien vérifier que le numéro de version du fichier **/workey-tomcat/version.properties** est correct.

Depuis la 6.11.0, les notes de versions sont publiées sur docs.efalia.com

Version 6-11-4 – 2021-MM-dd_hh-mm-ss_XX

Moteur

STalk

Designer HTML

Designer Flash / WorkeyOnAir

Application web

Workey App (v7)

- la valorisation d'un champ DropDown, après un refresh, est désormais réalisée (Redmine 10559)

Interface v6

WKYJS

API

Connecteurs et Agents

Console d’administration

- Augmentation de la taille des champs de paramétrage des agents.

Archivage

Recherche

Connecteur RestAPIConnector

- Auteur de l’article

Par Briac Date de l’article

- 2021-03-16
- Script de callback
  - Variables disponibles

Classe du connecteur : com.clog.workey.connectors.RestAPIConnector

Exécute une requête HTTP vers une WebService externe. Ses paramètres de lancement sont définis dans champ de formulaire contenant un objet JSON avec la structure suivante (sans les commentaires, seule la propriété “url” est obligatoire) : Lire la suite “Connecteur RestAPIConnector”

- Étiquettes
- Connecteur
