easySDI EXTRACT – Manuel d'exploitation

RÉFÉRENCE DU DOCUMENT

Référence : MAN_ASIT_Extract_ManuelExploitation_V1.9.docx
Version : 1.9
Date : 11 octobre 2022

HISTORIQUE DU DOCUMENT

Version	Date	Contenu	Créé par
1.0	26.10.2016	Instantiation	JLO
1.1	05.07.2017	Version initiale	YGR
1.2	18.09.2017	Documentation v1.0 application	YGR
1.3	13.11.2017	Modification charte graphique	YGR
1.4	18.09.2018	Modifications v1.1 application	YGR
1.5	24.09.2018	Ajout type champ règles -> TEXT	YGR
1.6	09.10.2018	Paramètres pour la taille des uploads	YGR
1.7	26.11.2019	Chemin de la config Tomcat sous Windows	YGR
1.8	08.03.2021	Taille cache Tomcat	YGR
1.9	23.08.2022	Mise à jour v2.0 (Passage à Java 17 et Spring Boot 2)	YGR

GUIDE DE LECTURE

1 INTRODUCTION

Ce document donne des informations concernant la gestion avancée de l'application EXTRACT.

2 CONFIGURATION DE L'APPLICATION

2.1 Fichier application.properties

Ce fichier contient le paramétrage de l'application à proprement parler. Les modifications sont prises en compte après le redémarrage de l'application Tomcat EXTRACT.

Il contient les propriétés suivantes :

application.external.url
URL permettant d'accéder à l'application. Cette valeur est notamment utilisée pour générer les liens dans les e-mails de notification.

email.templates.cache
Valeur booléenne définissant si les modèles de notifications par e-mail doivent être mis en cache.

email.templates.encoding
Encodage utilisé par les modèles de notification par e-mail

email.templates.path
Chemin relatif du répertoire contenant les modèles de notification par e-mail. Si le chemin est relatif au répertoire `WEB-INF/classes` de l'application, il doit être préfixé par `classpath:/`.

extract.i18n.language
Code de la langue utilisée par l'application. Un répertoire portant ce nom et contenant les fichiers de langue de l'application doit exister dans le répertoire `WEB-INF\classes\static\lang`. Si tel n'est pas le cas, il est possible de créer une nouvelle langue en se référant au chapitre 0 ci-dessous.

http.proxyHost
Nom du serveur à utiliser comme proxy pour les connexions externes. Cette propriété peut être omise si on n'utilise pas de proxy.

http.proxyPassword
Mot de passe permettant de s'identifier sur le serveur proxy. Cette propriété peut être omise si on n'utilise pas de proxy ou s'il ne demande pas d'authentification.

http.proxyPort
Port HTTP du serveur proxy. Cette propriété peut être omise si on n'utilise pas de proxy.

http.proxyUser
Nom de l'utilisateur permettant de s'identifier sur le serveur proxy. Cette propriété peut être omise si on n'utilise pas de proxy ou s'il ne demande pas d'authentification.

logging.config
Chemin du fichier contenant la configuration des logs de l'application. Si le chemin est relatif au répertoire `WEB-INF/classes` de l'application, il doit être préfixé par `classpath:`.

spring.batch.jdbc.initialize-schema
Chaîne définissant si l'initialiseur Spring Batch doit être démarré. Cette valeur doit être définie à `never` afin notamment d'éviter la création de tables inutilisées dans la base de données.

spring.batch.job.enabled
Valeur booléenne définissant si les tâches planifiées doivent être démarrées automatiquement par Spring Batch. Doit être défini à `false`. (Ces tâches sont démarrées par l'application.)

spring.datasource.driver-class-name
Nom qualifié de la classe du driver de base de données à utiliser.

spring.datasource.password
Mot de passe à utiliser pour se connecter à la base de données.

spring.datasource.url
Chaîne JDBC de connexion à la base de données.

spring.datasource.username
Nom de l'utilisateur permettant de se connecter à la base de données.

spring.http.multipart.max-file-size
Taille maximale de chaque fichier pouvant être uploadé dans l'application. Si on reçoit une erreur 500 lors de l'ajout d'un fichier volumineux dans le dossier d'une demande, il peut être utile d'augmenter cette valeur, en n'oubliant pas d'augmenter également `spring.http.multipart.max-request-size` si nécessaire.

spring.http.multipart.max-request-size
Taille maximale d'une requête d'upload. Cette valeur doit être supérieure ou égale à celle de `spring.http.multipart.max-file-size`.

spring.jpa.database-platform
Nom qualifié de la classe du dialecte à utiliser pour communiquer avec la base de données

spring.jpa.hibernate.ddl-auto
Valeur définissant l'attitude de l'application par rapport à l'architecture de la base de données. Par défaut, cette propriété est définie sur `update`, ce qui permet notamment de générer automatiquement les tables lors du premier démarrage de l'application. Par la suite, il est possible d'accélérer un peu le démarrage de l'application en définissant cette valeur à `none`. Il faut noter cependant que dans ce cas, si une table ou une colonne a été accidentellement supprimée, elle ne sera pas automatiquement recréée, ce qui peut entraîner des problèmes de fonctionnement. Il est également conseillé de repasser cette valeur à `update` lors de l'installation d'une nouvelle version d'EXTRACT afin que les éventuelles modifications de la base de données puissent être appliquées.

spring.jpa.properties.hibernate.id.new_generator_mappings
Valeur booléenne définissant s'il faut utiliser la dernière génération de générateur d'identifiants pour les entrées de la base. Doit être défini à `true`.

spring.jpa.show-sql
Valeur booléenne définissant s'il faut afficher le SQL généré par l'application, par exemple dans les logs. Il est fortement conseillé de laisser cette valeur à `false`.

spring.thymeleaf.cache
Valeur booléenne définissant si les modèles de pages de l'application doivent être mis en cache.

spring.thymeleaf.enabled
Valeur booléenne définissant s'il faut activer le moteur de templating. Doit être défini à `true`.

spring.thymeleaf.encoding
Encodage utilisé par les modèles de pages de l'application.

spring.thymeleaf.mode
Type de modèle utilisé pour les pages de l'application. Doit être défini à `HTML`.

spring.thymeleaf.prefix
Chemin du répertoire contenant les modèles de page de l'application. Si le chemin est relatif au répertoire `WEB-INF/classes` de l'application, il doit être préfixé par `classpath:/`.

spring.thymeleaf.suffix
Extension des modèles de page de l'application.

spring.thymeleaf.template-resolver-order
Priorité du moteur de templating pour la résolution des modèles de pages de l'application. Doit être défini à `1`.

table.page.size
Nombre de lignes à afficher dans les tableaux utilisant la pagination.

2.2 Fichier logback-spring.xml

Ce fichier permet de définir quelles informations doivent être écrites dans les logs de l'application. Les modifications sont prises en compte après le redémarrage de l'application Tomcat EXTRACT.

La configuration se passe en deux temps :

1. La définition d'un `appender`, soit un endroit vers lequel envoyer les informations à loguer, ainsi que le format de celles-ci. Il peut s'agir par exemple d'un fichier, d'une base de données, d'une console, etc.
2. La définition d'un `logger`, soit le lien entre une partie de l'application et un appender défini précédemment. Un logger peut utiliser simultanément plusieurs appenders, et un appender peut être utilisé par plusieurs loggers. Le nom du logger est le package dont les messages des classes doivent être traités. Cela inclus les éventuels sous-packages. Le logger définit également un niveau minimum de message à loguer. Le logger `root` définit les actions par défaut pour toutes les classes de l'application, y compris les dépendances.

Pour des informations plus détaillées, on peut se référer à l'aide de Logback.

2.3 Droits d'accès

Extract étant une application Tomcat, elle utilise l'utilisateur qui exécute le service Tomcat pour accéder aux différentes ressources (partages réseaux, dossiers locaux, etc.) Il est donc conseillé d'exécuter le service Tomcat avec un compte ayant accès aux ressources nécessaires.

2.3.1 Modifier l'utilisateur Tomcat sous Windows

1. Ouvrir le moniteur Tomcat (`tomcat8w.exe`, ou `Menu Démarrer > All Programs > Apache Tomcat > Configure Tomcat`)
2. Aller dans l'onglet `Log On`
3. Cocher `This account`
4. Entrer les informations du compte souhaité
5. Cliquer sur `OK`
6. Redémarrer le service Tomcat

2.3.2 Modifier l'utilisateur Tomcat sous Linux

1. Éditer le fichier `/etc/systemd/system/tomcat9.service`
2. Dans la catégorie `Service`, changer les valeurs des clés `User` et `Group` avec respectivement le nom de l'utilisateur et son groupe
3. Enregistrer le fichier
4. Redémarrer le service Tomcat

Le nom du fichier, son emplacement, voire la procédure entière peut s'avérer différente suivant la distribution utilisée, la version de Tomcat et son mode d'installation.

2.4 Optimisations

2.4.1 Taille du heap

L'exécution des scripts FME Desktop par EXTRACT peut entraîner une utilisation importante du heap. Si vous utilisez ce plugin, il est conseillé de définir la taille de départ du heap (paramètre `Xms`) à 1024 Mo et la taille maximale (paramètre `Xmx`) à au moins 2048 Mo.

La définition des ces paramètres se fait de la fichier `setenv.sh` (sous Linux) ou `setenv.bat` (sous Windows) qui se trouve dans le repertoire `bin` de Tomcat.

Si Tomcat fonctionne en tant que service Windows, ces paramètres se définissent dans l'onglet `Java` des propriétés du service. **Note :** Cet onglet n'est accessible qu'en passant par le moniteur Tomcat (`tomcat8w.exe`, ou `Menu Démarrer > All Programs > Apache Tomcat > Configure Tomcat`). Il n'est pas visible si l'on passe par la console de gestion des services Windows (`services.msc`).

Dans tous les cas, il faut ensuite redémarrer Tomcat pour que les changements soient pris en compte.

2.4.2 Locale par défaut de Tomcat

Dans les e-mails envoyés par Tomcat, les dates sont formatées en fonction du locale de Tomcat. Pour le définir, il faut utiliser les paramètres `Duser.language` et `Duser.region`.

Exemple :

```
-Duser.language=fr -Duser.region=CH
```

Ces paramètres se définissent de la même façon que pour la taille du heap (voir 2.4.1 ci-dessus).

2.4.3 Taille du cache

Au démarrage, il est possible que les logs contiennent une entrée indiquant qu'un élément n'a pas pu être chargé en raison de la taille du cache comme par exemple :

```
24-May-2017 10:52:57.029 WARNING [https-jsse-nio-8443-exec-10]
org.apache.catalina.webresources.Cache.getResource Unable to add the resource
at [/WEB-INF/classes/static/bower_components/select2/docs/_sass/vendor/font-
awesome/_variables.scss] to the cache for web application [/extract] because
there was insufficient free space available after evicting expired cache
entries - consider increasing the maximum size of the cache
```

Dans ce cas, on peut augmenter la taille maximale du cache en modifiant le fichier `context.xml` se trouvant dans le répertoire `conf` de Tomcat.

```
<Context>
    <!-- ... -->

    <!-- Larger resources cache for EXTRACT -->
    <Resources cacheMaxSize="102400" />

    <!-- ... -->
</Context>
```

3 AJOUT D'UNE NOUVELLE LANGUE

Il est possible de définir une nouvelle langue pour l'affichage de l'application EXTRACT. Pour cela :

1. Créer dans `WEB-INF\classes\static\lang` un nouveau répertoire ayant pour nom le code sur deux caractères de la langue désirée (par exemple `it`)
2. Copier le contenu du répertoire d'une langue déjà définie (fichiers `messages.properties` et `messages.js`)
3. Traduire les messages contenus dans ces fichiers

Le sélecteur de date utilisé sur la page d'accueil est également traduisible. Ses fichiers de langues sont situés dans le répertoire `WEB-INF\classes\static\lib\bootstrap-datepicker\locales`. Si la langue souhaitée n'est pas encore définie :

1. Copier un fichier de langue existant en changeant le code de langue dans son nom
2. Traduire les chaînes qu'il contient

Enfin, pour que l'application utilise la nouvelle langue, il faut modifier la valeur de la propriété `extract.i18n.language` dans le fichier `WEB-INF\classes\application.properties` et redémarrer l'application Tomcat.

4 PLUGINS

Le type de serveurs fournissant les demandes et les types de tâches pouvant être incluses dans un traitement sont gérés par des plugins qui peuvent être ajoutés, supprimés ou mis à jour indépendamment de l'application à proprement parler.

4.1 Installation ou mise à jour d'un plugin

Si un nouveau type de serveur ou de tâche est disponible, il suffit de placer son JAR dans le répertoire `WEB-INF/classes/connectors` (pour les types de serveurs) ou `WEB-INF/classes/task_processors` (pour les types de tâches) de l'application. En cas de mise à jour, il convient de supprimer le JAR de l'ancienne version afin d'éviter des conflits.

Il faut ensuite redémarrer l'application Tomcat EXTRACT pour que la modification des plugins soit prise en compte.

4.2 Développement d'un nouveau connecteur

- Le projet doit être un module Java et doit donc inclure un fichier module-info.java déclarant ses dépendances
- Le projet du nouveau connecteur doit définir une dépendance vers le projet `extract-interface`
- La classe principale du connecteur doit implémenter l'interface `IConnector`
- Le connecteur doit déclarer un constructeur sans paramètres
- Le code renvoyé par la méthode `getCode` doit être unique parmi les connecteurs EXTRACT
- La méthode `getParams` renvoie les paramètres du connecteur sous forme de tableau au format JSON. Si le connecteur n'accepte pas de paramètres, renvoyer un tableau vide : `[]`
- Les fichiers statiques utilisés par le connecteur (fichiers de propriétés, fichiers de langue, etc.) doivent être placés dans le sous-répertoire `resources/connectors/<code du plugin>/`
- Un fichier nommé `ch.asit_asso.extract.connectors.common.IConnector` doit être créé dans le sous-répertoire `resources/META-INF/services`. Son contenu doit être le nom qualifié de la classe principale du connecteur. Si cela n'est pas respecté, le plugin ne sera pas reconnu par EXTRACT.

4.3 Développement d'un nouveau plugin de tâche

- Le projet doit être un module Java et doit donc inclure un fichier module-info.java déclarant ses dépendances
- Le projet du nouveau connecteur doit définir une dépendance vers le projet `extract-interface`
- La classe principale du connecteur doit implémenter l'interface `ITaskProcessor`
- Le plugin de tâche doit déclarer un constructeur sans paramètres
- Le code renvoyé par la méthode `getCode` doit être unique parmi les plugins de tâche EXTRACT
- La méthode `getParams` renvoie les paramètres du plugin sous forme de tableau au format JSON. Si le plugin n'accepte pas de paramètres, renvoyer un tableau vide : `[]`
- La méthode `getPicto` doit retourner le nom de l'icône FontAwesome à afficher dans la barre de titre du plugin
- Les fichiers statiques utilisés par le plugin (fichiers de propriétés, fichiers de langue, etc.) doivent être placés dans le sous-répertoire `resources/plugins/<code du plugin>/`
- Un fichier nommé `ch.asit_asso.extract.plugins.common.ITaskProcessor` doit être créé dans le sous-répertoire `resources/META-INF/services`. Son contenu doit être le nom qualifié de la classe principale du plugin. Si cela n'est pas respecté, le plugin ne sera pas reconnu par EXTRACT.
- Le plugin de tâche est autorisé à modifier certains attributs de la requête originale qu'il renvoie avec le résultat. Ces attributs sont les suivants :

Attribut	Rôle
rejected	Valeur booléenne qui détermine si la requête peut être traitée par EXTRACT. Si cet attribut est défini à `true`, le traitement est interrompu et la demande est immédiatement exportée sans résultat vers son serveur d'origine. Dans ce cas, il est obligatoire de spécifier également une remarque explicative via l'attribut `remark`.
remark	Texte explicatif exporté avec la demande pour donner des informations à propos du traitement à celui qui a demandé la donnée. Sauf cas particulier, il est généralement recommandé d'ajouter sa remarque au contenu original de l'attribut afin de ne pas perdre d'information.

Si d'autres attributs de la requête sont modifiés, ils ne sont pas pris en compte.

5 PERSONNALISATION DE LA CARTE

Si vous souhaitez personnaliser les couches de la carte affichant l'emprise de la commande dans les détails d'une requête, cela peut être fait en définissant une configuration dans un fichier nommé map.custom.js placé dans le répertoire `WEB-INF\classes\static\js\requestMap`.

Ce répertoire contient un fichier `map.custom.js.dist` avec un exemple de configuration dont on peut partir si on le souhaite.

Les contraintes à respecter sont les suivantes :

- Le fichier `map.custom.js` déclare une fonction `initializeMap`. Cette méthode retourne une Promise qui se résout avec la carte
- L'attribut `target` de la carte doit avoir pour valeur la chaîne `orderMap`
- Chaque couche doit posséder dans ses options un attribut `title` dont la valeur est le libellé à afficher dans le widget de liste des couches
- Les couches contenant un fond de plan doivent contenir dans leurs options un attribut `type` ayant pour valeur la chaîne `base`

6 LIMITATIONS DU CHAMP CONTENANT LES RÈGLES

La création des champs de type texte étant problématique lorsqu'on la traite uniquement avec JPA (afin que l'application ne soit pas liée à un système de base de données en particulier), il a été décidé de créer en VARCHAR le champ qui contient les règles associant les demandes à un traitement. Malheureusement, cela implique une limitation du nombre de caractères de chaque règle à 65'000.

Avec PostGreSQL, Il est cependant possible de supprimer cette limitation en changer manuellement le type du champ `rule` de la table `rules`. La commande SQL pour effectuer cette modification est la suivante :

```
ALTER TABLE rules ALTER COLUMN rule TYPE TEXT
```

Le contenu du champ pour les enregistrements existants est conservé.

Il n'est pas garanti qu'une modification similaire fonctionne avec un autre SGBD.