Hot folder exercise

Step 1:

Fortunately, the accelerator comes armed with an initial and complete configuration of the hot folder.

The ImpexConverters for prices, products, medias, stocks, customers…, are already there and good to go.

However in this article, we will try to extend the hot folder functionalities to be able to import an other item type **UnitModel**.

To achieve this you need to define three setups

1. Define a base directory where the csv files will be put.
2. Initiate your flow with the catalog and base directory…
3. Create an ImpexConverter and associated with a MappingConverter.

Step 2:

Create an xml file hot-folder-store-training-spring.xml inside the …\hybris\bin\custom\training\trainingcore\resources\trainingcore\integration

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:int="http://www.springframework.org/schema/integration"
           xmlns:file="http://www.springframework.org/schema/integration/file"
           xmlns:p="http://www.springframework.org/schema/p"
           xmlns:context="http://www.springframework.org/schema/context"
           xsi:schemaLocation="http://www.springframework.org/schema/integration
         http://www.springframework.org/schema/integration/spring-integration.xsd
         http://www.springframework.org/schema/integration/file
         http://www.springframework.org/schema/integration/file/spring-integration-file.xsd
         http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context.xsd">

  <context:annotation-config/>
</beans>
```

Step 3:
Import the hot-folder-store-training-spring.xml file to the trainingcore-spring.xml

```
<!-- Spring Integration -->
<import resource="classpath:/trainingcore/integration/hot-folder-store-training-spring.xml"/>
<import resource="classpath:/trainingcore/integration/hot-folder-store-electronics-spring.xml"/>
<import resource="classpath:/trainingcore/integration/hot-folder-store-apparel-spring.xml"/>
<import resource="classpath:/trainingcore/integration/hot-folder-common-spring.xml"/>
```

Step 4 :Inside the hot-folder-store-training-spring.xml add a base directory and an inbound-channel-adapter to watch over the base directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns... >

   <context:annotation-config/>

        <!-- Config a base directory -->
        <bean id="baseDirectoryTraining" class="java.lang.String">
                <constructor-arg value="#{baseDirectory}/${tenantId}/training" />
        </bean>

        <!-- Scan for files inside the base directory with names matches the pattern ^(.*)-(\d+)\.csv
-->
        <file:inbound-channel-adapter id="batchFilesTraining" directory="#{baseDirectoryTraining}"
                                                                        filename-regex="^(.*)-(\d+)\.csv"
                                                                        comparator="fileOrderComparator">
                <!--  Periodic trigger in milliseconds -->
                <int:poller fixed-rate="1000" />
        </file:inbound-channel-adapter>
</beans>
```

Note : The #{baseDirectory} by default is ${HYBRIS_DATA_DIR}/acceleratorservices/import, you can re-define it with the property : acceleratorservices.batch.impex.basefolder

Step 5: Add an outbound-gateway to move the received file to processing and invoke the first step of the flow.

```
<!-- ...\hybris\bin\custom\training\trainingcore\resources\trainingcore\integration\hot-folder-store-
training-spring.xml -->

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns... >

   <context:annotation-config/>

        ...

        <!-- Move the file to processing and start the flow -->
        <file:outbound-gateway request-channel="batchFilesTraining" reply-
channel="batchFilesTrainingProc"
                                                    directory="#{baseDirectoryTraining}/processing"
                                                    delete-source-files="true" />

</beans>
```

Step 6 :Create a service-activator which is the first step of the flow, it feeds the flow with the catalog and other relevant information.

```
<!-- ...\hybris\bin\custom\training\trainingcore\resources\trainingcore\integration\hot-folder-store-
training-spring.xml -->
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns... >

   <context:annotation-config/>

         ...

         <!-- Initialize the batch header with relevant information -->
         <int:service-activator input-channel="batchFilesTrainingProc" output-
channel="batchFilesHeaderInit"
                                                      ref="trainingHeaderSetupTask"
                                                      method="execute" />

         <bean id="trainingHeaderSetupTask"
class="de.hybris.platform.acceleratorservices.dataimport.batch.task.HeaderSetupTask">
                 <property name="catalog" value="trainingProductCatalog" />
                 <property name="net" value="false" />
                 <property name="storeBaseDirectory" ref="baseDirectoryTraining" />
         </bean>
</beans>
```

Step 7: Create ImpexConverter and ConverterMapping

a. Create an ImpexConvert for the UnitModel.

The ImpexConverter has two properties a header and an impexRow :

header : will be the header of the generated impex.
impexRow : defines the mapping between the column of the csv file and the impex file.
Inside the hot-folder-store-training-spring.xml, create a Spring bean from DefaultImpexConverter, with :

```
header : INSERT_UPDATE
Unit ;unitType[unique=true] ;code[unique=true] ;name[lang=$lang] ;conversion
impexRow :  ;{+0}  ;{+1}  ;{2}  ;{3}
```

```
<!-- ...\hybris\bin\custom\training\trainingcore\resources\trainingcore\integration\hot-folder-store-
training-spring.xml -->

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns... >
   <context:annotation-config/>
         <!-- UnitModel impex converter -->
         <bean id="unitConverter"
class="de.hybris.platform.acceleratorservices.dataimport.batch.converter.impl.DefaultImpexConver
ter">
                 <property name="header">
                         <value>
                                 #{defaultImpexProductHeader}
INSERT_UPDATE Unit;unitType[unique=true];code[unique=true];name[lang=$lang];conversion
                         </value>
                 </property>
                 <property name="impexRow">
                         <value>;{+0};{+1};{2};{3}</value>
```

```
                </property>
        </bean>
</beans>
```

This means that :

{+0} means that column 0 will be mapped to unitType[unique=true]
{+1} means that column 1 will be mapped to code[unique=true]
{2} means that column 2 will be mapped to name[lang=$lang]
{3} means that column 3 will be mapped to conversion

b.Map the unitConverter to a file name prefix using DefaultConverterMapping.

```
<!-- ...\hybris\bin\custom\training\trainingcore\resources\trainingcore\integration\hot-folder-store-training-spring.xml -->

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns... >
   <context:annotation-config/>
        <!-- UnitModel impex converter mapping to unit prefix -->
        <bean id="unitConverterMapping"

class="de.hybris.platform.acceleratorservices.dataimport.batch.converter.mapping.impl.DefaultConverterMapping"
                p:mapping="unit"
                p:converter-ref="unitConverter"/>
</beans>
```

This means that all the csv files start with unit will be treated using the unitConverter, example unit-fr-1234567.csv.

c.Run an ant all command then restart the server and you are good to go.

Step 8: Hot Folder in Action

To test what you have accomplished so far, put a csv file that respect the impexRow format and the p:mapping="unit" inside the : ${HYBRIS_DATA_DIR}\acceleratorservices\import\master\training

Example : unit-en-123456789.csv

```
toto,    toto,    toto,    1
```

If everything goes well a Unit instance with code toto and name toto should be created.