**Nouveau metadata\_delta\_finder**

* 1 custom transformer (METADATA\_DELTA\_FINDER\_GEN.fwx) et 1 script python appelé par le pythonCaller\_7 (METADATA\_DELTA\_FINDER\_GEN.py)
* Le custom transformer (CT) METADATA\_DELTA\_FINDER\_NG.fwx comparait les valeurs de certains attributs (id,record\_last\_modified,resources{}.url,title,notes, resources{}.name) entre les attribut dans le pycsw ou local vs les nouvelles valeurs des api des provinces; afin d’évaluer si les métadonnées étaient inchangées, modifiées, insérées ou supprimées. Ce transformer se basait sur le fait que si le record\_last\_modified des provinces était modifié les métadonnées étaient modifiées. Par contre, pour certaines provinces cet attribut était un timestamp lorsqu’on appelait l’api; et pour d’autres il n’y avait pas d’attributs lié aux modifications de métadonnées (on prenait donc les attributs liés aux modifications des données).
* On a donc décidé de modifier ce CT afin de comparer tous les attributs obligatoires du HNAP (pour le géo) et CKAN (pour le non-géo) sauf le ‘record\_last\_modified’.
* 1 fichier yaml par province (1 pour le géo et 1 pour le non-géo) contient la liste des attributs de référence (pycsw ou JSON/XML locaux) et ceux de l’api de la province (nom des attributs des workspaces des provinces avant le CT metadata\_delta\_finder); selon :

attr\_1:

attr\_name: title

attr\_ref: MD\_Metadata.identificationInfo.MD\_DataIdentification.citation.CI\_Citation.title.CharacterString

attr\_new: title

ou : attr\_1 n’est que l’attribut 1 (seul l’attribut ‘id’ doit se nomme id puisqu’on l’utilise dans le script python, pour les autres on peut les appeler n’importe comment); attr\_name est le nom qu’on veut donner à l’attribut (on peut mettre n’importe quel nom, ce n’est que pour les identifier en cas de différence ou d’erreurs); et attr\_ref et attr\_new correspondent aux noms des attributs de référence vs des provinces.

*Pour l’instant, le yaml doit être inséré comme paramètre du CT; par contre on devrait améliorer pour qu’il puisse lire directement le fichier yaml dans les lookuptable de chaque province.*

* Si un attribut n’a que des valeurs nulles; il ne faut pas l’inclure puisqu’il n’apparait pas dans la liste des noms d’attributs (voir pythonCaller\_2 et pythonCaller\_3 du CT). Ces listes sont utilisées pour voir si les attributs new et ref nommés dans le yaml (attr\_ref et attr\_new) existent ou non. Sinon le CT plante et log l’erreur. Si certaines valeurs sont nulles parmi toutes les métadonnées ce n’est pas un problème (elles vont juste être changées pour ‘NULL’ dans le script python).

*Une amélioration au CT serait de trouver comment faire pour pouvoir inclure les attributs n’ayant que des valeurs nulles.*

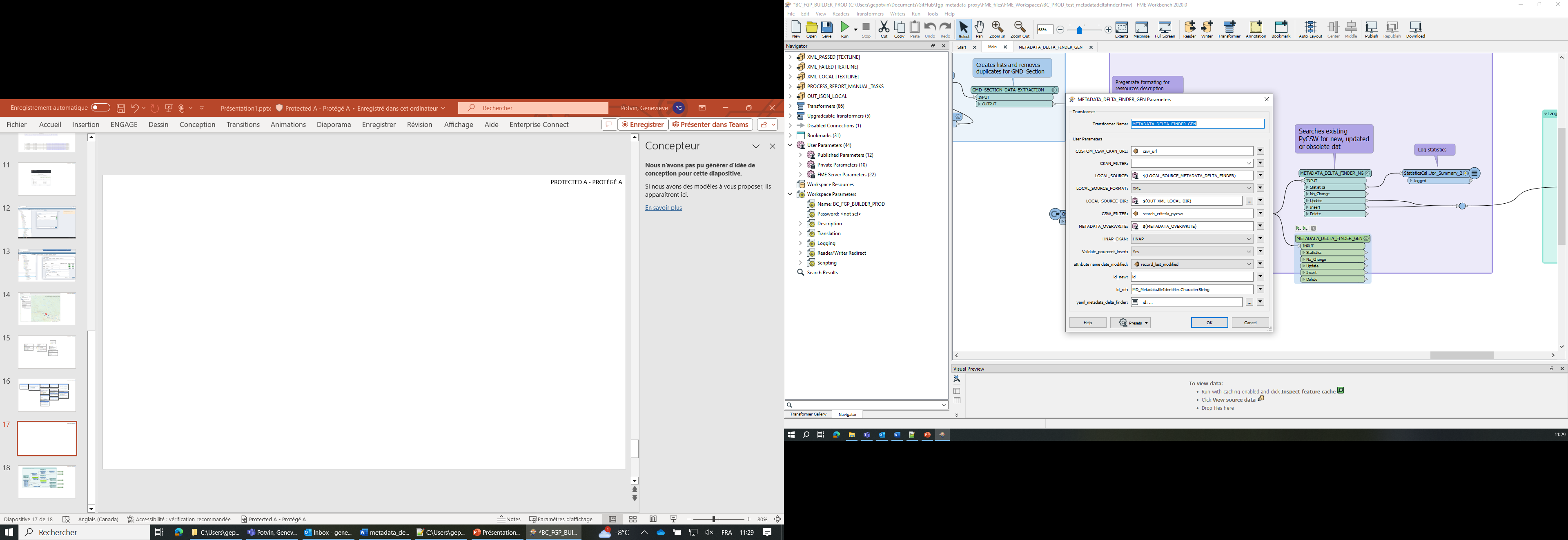
* Les attributs et les listes peuvent être traitées par ce CT; par contre pour les listes tous les éléments de la liste vont être triés puis joints pour en faire 1 attribut.
* Si le nom des attributs contiennent des nombres, il faut les renommer pour enlever ces nombre (voir le pythonCaller\_2 et le pythonCaller\_3). Par contre, si on veut prendre uniquement l’index 0 d’une liste, comme on ne peut pas mettre le ‘0’ on peut juste mettre la liste si les autres éléments sont nuls; le CT ne va traiter que l’élément non-null (mais attention s’il y a des valeurs dans les autres index; mieux vaut renommer l’attribut).
* exemple d’erreur et pistes de solution :

*ERROR: attribute: MD\_Metadata.contact.CI\_ResponsibleParty.role.CI\_RoleCode.codeListValue (yaml\_name: contact\_role) does not exists. Revise attribute name in the yaml file*

Soit l’attribut MD\_Metadata.contact.CI\_ResponsibleParty.role.CI\_RoleCode.codeListValue n’existe pas (pas le bon nom d’attribut), soit toutes ces valeurs sont vides. Parfois il existe pour certaines métadonnées mais pas d’autres, alors il faut renommer le bon attribut (les attributs ref ont souvent ce genre de problèmes puisqu’il ajoute des {} lorsqu’il y a plusieurs index mais pas lorsqu’il n’y en a qu’un seule. Il faut donc ajouter {0} lorsqu’il n’y a qu’une seule valeur dans la ‘liste’ : voir attributeRenamer\_3 dans le CT).

**À faire pour terminer :**

* Les yaml pour le géo des provinces sont fait (ils sont dans les lookuptable de chaque province); mais devraient tout de même être validés avant de modifier tous les workspaces.
* Les yaml pour le non-géo n’ont pas encore été créés. Il faut d’abord faire la liste des attributs obligatoires et créer les yaml selon la structure du géo.
* Pour valider que tout est correct : regarder la liste des attributs différents (soit dans le translation log ou l’attribut delta\_attr{} après le pythonCaller\_7 du CT (AttributeExposer\_6)
* AttributeExposer\_6 :enlever record\_last\_modified. le remplacer par le paramètre date\_modified puisqu'il peut s'appeler différement d'une province à l'autre
* Dans le script python (METADATA\_DELTA\_FINDER\_GEN.py) à la ligne 70 : identifie les coordonnées par ‘bound\_l’ : trouver une autre manière d’identifier les coordonnées (peuvent être appelées différemment d’une province à l’autre)
* Dans le script python (METADATA\_DELTA\_FINDER\_GEN.py) à la ligne 73 : renommer attr\_update2 et attr\_update3 pour que ce soit plus compréhensible
* Finalement, modifier les workspaces de chaque province pour remplacer le METADATA\_DELTA\_FINDER\_NG.fwx par le nouveau METADATA\_DELTA\_FINDER\_GEN.fwx. ajouter un paramètre (published parameters) pour validate\_pourcent\_insert et mettre à yes par défaut.

****