Ajouter un module à la synchronisation

Last edited by **Maelan LE BORGNE** 4 years ago

Une fois la base de données mise à jour avec le modèle du nouveau module, voici ce qu'il convient de faire pour l'intégrer à la synchronisation.

Param

- Créer une classe SettingsMonnouveaumodule.php dans *app/models/EpackData/EpackParams*, qui hérite de `EpackData\EpackParam`.
- Dans les annotations de la classe, faire figurer l'annotation `@XML(Name="DefMonNouveauModule", FileName="MonNouveauModule")`. L'attribut Name correspond au nom des éléments XML et FileName correpond au nom du fichier XML. Le fichier XML généré ressemblerai donc à

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfDefMonNouveauModule xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
  <DefMonNouveauModule>...</DefMonNouveauModule>
</ArrayOfDefMonNouveauModule>
```

- Ajouter une propriété à la classe pour chaque champs dans la table associée. Attention :
  - La colonne de clé primaire (id_settings_MonNouveauModule) doit être associée à la propriété `$idServer`. Pour que le mapping se fasse tout de même, ajouté l'attribut *column* à l'annotation *@Column* : `@Column(column="id_settings_MonNouveauModule", type="integer", length=11, nullable=false)`.
  - Si un champ doit être serialisé en XML, ajouter l'annotation `@XML(name="nomDeLelementXML")`
  - Si besoin, ajouter l'attribut *function* ou *multiArgs* à l'annotation *@XML* pour effectuer un traitement lors de la Serialisation/Deserialisation. Par exemple, un champs booléen est stocké en base sous forme d'entier 1 ou 0, et en XML en "true" ou "false". On ajoute donc *function="boolToString"* à l'annotation *@XML* pour convertir la valeur de ce champs. Pour plus d'informations sur les fonctions disponibles et l'implémentation de nouvelles fonctions, voir Serialisation.
  - Si la deserialisation de ce module nécessite un traitement plus complexe, il est possible de surcharge la méthode `convertFromXml($function_mapping)`
- Définir la table à laquelle est liée cette classe dans la fonction `getSource()`
- Si le modèle du module est composé de plusieurs tables, les relations se définissent la fonction `initialize()` (voir doc phalcon ou exemple dans le code existant)

Logs

- Créer une classe Tbmonnouveaumodule.php dans *app/models/EpackData/EpackLogs*, qui hérite de `EpackData\EpackLog`.
- Dans les annotations de la classe, faire figurer l'annotation `@CSV(Name="TbMonNouveauModule")`. L'attribut Name correspond au nom de la table dans la base Access de la solution ePack Hygiene.
- Ajouter une propriété à la classe pour chaque champs dans la table associée. Attention :
  - Si besoin, ajouter l'annotation *@CSV* avec l'attribut *function* pour effectuer un traitement lors de la Serialisation/Deserialisation. Par exemple, un champs booléen est stocké en base sous forme d'entier 1 ou 0, et en XML en "true" ou "false". On ajoute donc *function="boolToString"* à l'annotation *@CSV* pour convertir la valeur de ce champs. Pour plus d'informations sur les fonctions disponibles et l'implémentation de nouvelles fonctions, voir Serialisation.
  - Si la deserialisation de ce module nécessite un traitement plus complexe, il est possible de surcharge la méthode `convertFromCSV($function_mapping)`
- Définir la table à laquelle est liée cette classe dans la fonction `getSource()`

Finalisation

Les params des modules ne synchronisent pas tous de la même manière. Certains ne font que descendre du manager, certains ne font que remonter de la solution. Si les créations/modifications de param fait sur la solution **doivent se synchroniser** sur le manager :

- Ajouter le nom du XML dans dans app/config/list_sync_params_up.yaml **Sinon **:
- Ajouter le nom de la classe à la constante `NO_SYNC_UP_PARAMS` de app/plugins/Helpers/EntityHelper.php

Si les les params ne sont ** pas modifiables sur le manager** mais doivent remonter de la solution :

- Ajouter le nom de la classe à la constante `NO_SYNC_DOWN_PARAMS` de app/plugins/Helpers/EntityHelper.php

Si le modèle de **param** du module est composé de **plusieurs tables** (relations su id_settings_***) :

- Une seule classe doit se synchroniser et les autres doivent être ajoutées à la constante `NO_SYNC_PARAMS`
- La purge des ces params doit être empechée en ajoutant le nom des classes à la variable `$exclusion_list` de la fonction `getParamsToPurge()` dans app/plugins/Helpers/EntityHelper.php

Si le modèle de **logs** du modules sont composés de plusieurs tables (relations sur Id) :

- Ajouter le nom des classes aux tableaux d'exclusion des fonctions removeIds() et reorderId() dans app/plugins/Managers/DataManager.php