ExtensionsCustomisations
Extensions and Customisations

Updated Jul 13, 2010 by kyle.br...@gmail.com

If the HIT Isn't Meeting Your Needs

The HIT was designed to be as extensible and customisable as possible, so that support for new protocols, encoding, etc could be added.

Adding Extensions

The HIT can be extended in a number of ways:

1. You could add a new type of Harvester. The HIT uses separate Harvesters to speak DiGIR, BioCASE, TAPIR, and DwC-Archive. Because each protocol differs in the way information can be requested and returned, it's convenient to treat them separately, and standardise how data gets stored (in text files). Should a new protocol be added, it would be relatively easy for a Java coder to write the classes and plug it in.

2. You could add a new type of Encoding. Each protocol could be exchanging data encoded in DwC or ABCD for example. Of interest to the HIT, is which DwC or ABCD XPath corresponds to which term that we're interested in. Simple mapping files for each encoding manage these correspondences. Should a new encoding be added for some protocol, it is relatively simple to create a new mapping file and plug it in.

3. You could add a new Synchroniser. A synchroniser synchronises (indexes) data stored in text file(s) into a database. One such Synchroniser has been written so far, that synchronises data into the GBIF Portal database. Because each database might have a different structure, this GBIF Synchroniser won't work for everybody. Ultimately, someone wanting to synchronise into their own database structure would have to either adapt the GBIF one, or create their own in Java.

Making Customisations

The HIT can be customised in one way:

1. You could customise which terms get parsed for, or in other words, the data (in text file(s)) that gets stored. The simple mapping files mentioned as part of an Encoding can be manipulated. For DwC for example, the 'XPath to term-of-interest' mappings can represent the full list of terms, or it can represent only the minimum number of terms that are required. For example, GBIF is only interested in about 40 terms from DwC. Therefore for someone interested in storing different data from GBIF, like only geospatial data, all they would have to do is customise the mapping files for the different encodings.

Adding a New Harvester

Convert registered AccessPoints from UDDI as BioDatasources

Essentially, every AccessPoint registered for a data publisher in the UDDI gets converted into a separate BioDatasource. That AccessPoint's type should correspond to a type of HarvesterFactory.

For example, the AccessPoint of Academy of Natural Sciences http://digir.ansp.org/digir/DiGIR.php has type = "DiGIR"

At the time of writing there were currently five different types of HarvesterFactories that the HIT is capable of handling. New AccessPoint type - HarvesterFactory class mappings get added to the following map:

```
harvesterFactoryMap.put("BioCASe", "org.gbif.harvest.biocase.BiocaseMetadataFactory");
harvesterFactoryMap.put("DiGIR", "org.gbif.harvest.digir.DigirMetadataFactory");
harvesterFactoryMap.put("MaNISDiGIR", "org.gbif.harvest.digir.DigirMetadataFactory");
harvesterFactoryMap.put("TAPIR", "org.gbif.harvest.tapir.TapirMetadataFactory");
harvesterFactoryMap.put("DarwinCoreArchive", "org.gbif.harvest.dwcarchive.DwcArchiveHarvesterFactory");
```

To remain consistent with the way that other protocols have been developed, one needs to create both a metadata updater (MetadataHandler/MetadataFactory) and an operator (Harvester/HarvesterFactory).

Create a Metadata Updater

Note, the following work all takes place inside project harvest-webapp

- Create a MetadataFactory which extends from extends AbstractHarvesterFactory and implements ApplicationContextAware. A MetadataFactory creates a MetadataHandler and provides enough medatadata about it for its UI integration.
  - Add the I18N key values (where necessary) to the ApplicationResources.properties.
  - Register the Factory inside the applicationContext.xml, adding an appropriate bean to the list with id="harvesterFactories".
  - Create a new bean corresponding to this MetadataHandler
  - Remember to create and use an instance of a I18nLogFactory for logging

- Create a MetadataHandler which implements Harvester.
  - Create an entry point method for each registered method you want to be visible from the UI (each of these methods has to be added to the list of operations returned in the MetadataFactory's getOperations()). Such methods should throw Exceptions of type HarvesterException.
    - Remember to create and use an instance of a I18nLogFactory for logging
- Adjust UI to accommodate new MetadataHandler
  - Modify /scripts/gbif/datasources.js so that the ExtJs grid formatting is in line with the other MetadataFactories.
  - Modify /WEB-INF/pages/list.jsp so that the name-ranges link never appears, as this is only for Harvesters.
  - Modify /WEB-INF/pages/edit.jsp so that the MetadataFactory appears in the Harvester Factories drop-down both when in both add and edit modes.

Adding a New Encoding

For each protocol, the HIT supports the most common encodings.

For TAPIR 1.0:

- DwC 1.0
- DwC 1.4
- DwC 1.4 Geospatial
- DwC 1.4 Curatorial
- ABCD 1.2
- ABCD 2.06

For DiGIR 1.0:

- DwC 1.0
- DwC 1.4
- DwC 1.4 Geospatial
- DwC 1.4 Curatorial

For MaNIS DiGIR 1.0:

- (MaNIS) DwC 1.0
- (MaNIS) DwC 1.21

For BioCASE 1.3:

- ABCD 1.2
- ABCD 2.06

CASE 1. We want to add support for a new TAPIR 1.0 encoding?

We recently had to do this for one TAPIR publisher, who chose to encode their data using DwC 1.2. To extend the HIT for TAPIR (1.0) to be able to support this encoding is really simple. Below are the steps. (Note that instructions for other protocols would be similar, but not exactly the same)

1. write new property mapping file: indexMapping_dwc_1_2.properties (the name should follow the standardised format used for other mapping files, and should be added to the appropriate folder, i.e. harvester-tapir project's /src/main/resources/org/gbif/harvest/tapir/tapir_1_0/mapping. see here for an example mapping file)
2. write new search template (unique to TAPIR. see here for an example)
3. update appropriate template mapping file for newly created search template (unique to TAPIR. i.e. searchTemplateMapping.properties inside harvester-tapir project's /src/main/resources/org/gbif/harvest/tapir/)
4. update conceptualMapping.properties file with newly created property mapping file (located in harvest-webapp project's: /src/main/resources/org/gbif/harvest/tapir/mapping. This is most important, as this is where the HIT looks to know which property mapping file to use)
5. update outputModelMapping.properties file (unique to TAPIR. located in harvest-webapp project's: /src/main/resources/org/gbif/harvest/tapir/mapping)

Should the encoding be ABCD, the instructions will slightly differ - please contact the developer for help.

CASE 2: We want to add support for a new MaNIS DiGIR 1.0 conceptual schema.

Recently one of our MaNIS DiGIR publishers updated to a new conceptual schema (with new conceptual schema location) based on DwC 1.21. The HIT relies on the conceptual schemas to identify which protocol and which encoding If the HIT doesn't recognise a conceptual schema, then it will use the default protocol and encoding. For DiGIR, these would be DiGIR 1.0 and DwC 1.0. Therefore, it's important to update HIT's (DiGIR) mapping files so that support for this conceptual schema can be added, and so that BioDatasources created will harvest data using the appropriate DwC (MaNIS) 1.21.

So, first we need to ask 2 questions:

1. Does the HIT already support the MaNIS DiGIR 1.0 protocol?
2. Does the HIT already support the DwC (MaNIS) 1.21 encoding?

Because the answer to both these questions is YES, it's really easy to add support. All we have to do is add a single line to two files:

1. protocolMapping.properties (manages schema to protocol mappings, where the key is the schema location, and the value is the HIT formatted protocol name: MaNIS DiGIR 1.0 -> manis_digir_1_0)

```
#-- "MaNIS 1.21" mappings -- Notice colon in URL must be escaped
.
.
http\:\//bnhm.berkeley.edu/manis/DwC/darwin2jrw100317.xsd=manis_digir_1_0
```

2. conceptualMapping.properties (manages schema to encoding mappings, where the key is the schema location, and the value is the HIT formatted encoding name: DwC 1.21 -> indexMapping_manis_dwc_1_21)

```
#-- "MaNIS 1.21" mappings -- Notice colon in URL must be escaped
.
.
http\:\//bnhm.berkeley.edu/manis/DwC/darwin2jrw100317.xsd=indexMapping_manis_dwc_1_21
```

If the answer, however, was NO to both these questions, there's a few additional steps in addition to the ones above that we'd have to take.

3. Create the directory with the HIT formatted protocol name "manis_digir_1_0" inside the directory:

```
harvest-webapp/src/main/resources/org/gbif/harvest/digir
  >manis_digir_1_0
```

4. Create a new directory inside the directory /manis_digir_1_0 called "mapping" and inside here add a new property mapping file with the HIT-formatted encoding name: indexMapping_manis_dwc_1_21.properties (see here for an example)

```
harvest-webapp/src/main/resources/org/gbif/harvest/digir
  >manis_digir_1_0
    >mapping
      >indexMapping_manis_dwc_1_21.properties
```

5. Create a new directory inside the directory /manis_digir_1_0 called "template" and inside here add new search templates (see here for an example)

```
harvest-webapp/src/main/resources/org/gbif/harvest/digir
  >manis_digir_1_0
    >mapping
    >template
      >constants.vm
      >filter.vm
      >header.vm
      >inventory.vm
      >postamble.vm
      >preamble.vm
      >search.vm
```

Adding a New Synchroniser