hale Workshop
Introduction to new features

22.05.2018
Agenda (1/2)

hale studio
- Project variables
- Improved alignment migration
- hale connect integration
- Spatial Index / Join
- Split GML by feature type
- SQL queries as data source
Agenda (2/2)

hale Command Line Interface
- Merging alignments
- Custom success conditions for transformations
- Rewrite data

Plugins
- GeoServer AppSchema: Support for isolated workspaces
- XtraServer configuration Import & Export
hale studio
Project variables (3.0)

Use case:
- Make variables in a transformation configurable for individual transformation runs

How to use?
- Configure project variables (default values) in the transformation project
- Use them in transformation functions (e.g. Assign, Formatted String, Groovy scripts)
- Override using environment variables or system properties
Improved alignment migration (3.4)

Use case:
- Change / update source or target schema (e.g. to a new version)
  - Examples:
    - Migrate alignment from one INSPIRE version to the next
    - Updated alignment to changed source schema

How to use?
- Use „Reload and update schema“ (replace target schema) or
- Import existing alignment into a new project or
- „Merge“ two alignments (see example shown later)
hale connect integration (3.3)

Directly interact with hale connect from hale studio:
- Load and save projects
- Export projects for sharing
Spatial Index / Join (3.3)

Use case:
– Find objects that are related spatially to a certain geometry / object

How to use?
– Using the Spatial Join transformation function or
– Using the `spatialIndexQuery` helper function in Groovy scripts
Split GML by feature type (3.4)

GML output can be split by feature types
- One file per feature type
- References across feature types are adapted to include the file name

Note: In hale studio 3.4 there is a bug regarding the naming of the created files.
SQL queries as data source (3.3)

Use case:

– Support transformation from database side (but views can or shall not be used)

– Parameterisable queries via project variables (e.g. dynamically adjust WHERE clause)
SQL queries as data source (3.3)
SQL queries as data source (3.3)

SQL Query

Successfully tested query. It yields a result with 2 columns.

Query name: SQL_Query_Type

```
SELECT cr.MUDAB_REF_CR AS REF_NO,
     p.MEDIUM_PAR AS Datenart
FROM mudabadm.cruise cr
JOIN mudabadm.qsinternal qsi ON (cr.CRUISEID = qsi.CRUISEID)
JOIN mudabadm.parameter p ON (qsi.PARAMETERID = p.PARAMETERID)
WHERE cr.mudab_ref_cr = 20160198
ORDER BY 1, 2
```
SQL queries as data source (3.3)

SQL Query

Successfully tested query. It yields a result with 2 columns.

Query name: SQL_Query_Type_withVariable

```sql
SELECT cr.MUDAB_REF_CR AS REF_NO,
    p.MEDIUM_PAR AS Datanart
FROM mudabadm.cruise cr
JOIN mudabadm.qsinternal qsi ON (cr.CRUISEID = qsi.CRUISEID)
JOIN mudabadm.parameter p ON (qsi.PARAMETERID = p.PARAMETERID)
WHERE cr.mudab_ref_cr = {{project:mudabRef}}
ORDER BY 1, 2
```

Test query
Merging alignments (3.4)

Combine two hale alignments

A ➞ B ➞ C

A ➞ C

#15 22.05.2018

www.wetransform.to
Merging alignments (3.4)

Combine two hale alignments

Current restriction: no inherited mappings supported
Merging alignments (3.4)

Command in hale-cli

```
$ hale project merge --help
usage: hale project merge [options] ...

--help
--log-err <file>
--log-out <file>
--log-reports <file>
--matching-project <file-or-URL>
The project defining a schema matching
--source-project <file-or-URL>
The source project to migrate
--statistics <file>
File to write merge statistics to
--target <file>
Target project file
```
Merging alignments (3.4)

Result:
- Combined project/alignment
- Necessary changes can only be done automatically in part
- Manual verification/changes required
  - Support via hints where changes may be necessary
Custom success conditions for transformations (3.4)

Use case:
- Specific conditions when a data set should be further processed after transformation
  - Example: Transformation is only considered successful if
    - XML Schema validation was run and yielded no errors / warnings
    - At least one object (of a certain type?) was created during the transformation
    - No errors occurred during the transformation

Note: The default behavior for transformation success (return code zero) is that the transformation completed (errors may have occurred), a result was written and validations on the written data were successful.
Custom success conditions for transformations (3.4)

How to use?
– Specify Groovy script to be evaluated on transformation statistics or
– Process transformation statistics (Json) with a tool of your choice

Options for transformation command line:
-succesEvaluation <file-or-URI-to-script>
-statisticsOut <statistics-file>
Rewrite data (3.3.1)

Use case:
- Rewrite data with hale without the need to rerun a transformation
  - Examples:
    - Apply a coordinate transformation
    - Split output in multiple files
    - Do an upload via WFS-T interface

How to use?
```
hale data rewrite --help
```
Plugins
GeoServer AppSchema: Support for isolated workspaces (3.4)

- Support for isolated workspaces (multiple workspaces for the same namespace may exist)
- Requires GeoServer 2.13 (+ AppSchema plugin)
XtraServer configuration Import & Export (3.4)

– Requires XtraServer by interactive instruments
– Import of XtraServer mapping configuration (Database to XML/GML)
– Export to XtraServer mapping configuration (limited set of functions)
Questions?