

Custom Project Specific Algorithm Handlers

Custom Project Specific Algorithm Handlers

The project specific handlers define behavior specific to one project. A template project handler named `customProjectSpecificHandler.java` is provided in the `mapping-custom` Maven project.

When writing a custom handler, the following types of behavior should be considered:

- Processing Workflow Actions
- Automatic computation of mapping parameters
- Validating Map Records
- Release-specific validation and methods

Each category is described below. For each method, the default behavior is described.

If additional methods are required, they should either be folded into one of the following methods or added to the supertype `DefaultProjectSpecificHandler` on a forked version of the Mapping Tool GitHub project.

Processing Workflow Actions

For general information on workflow, see [Editing](#), [Tracking](#), [Workflow](#).

If using the built-in workflow paths, no customization is required. If using a [custom workflow path](#), behavior must be specified for any workflow action used in the custom `WorkflowPathHandler`.

Workflow Methods

```
public Set<MapRecord> assignFromScratch(...)
public Set<MapRecord> assignFromInitialRecord(...)
public Set<MapRecord> cancel(...)
public Set<MapRecord> finishEditing(...)
public Set<MapRecord> publish(...)
public Set<MapRecord> saveForLater(...)
```

Each method takes the following arguments:

- `TrackingRecord trackingRecord`: The `TrackingRecord` for the concept and project
- `Set<MapRecord> mapRecords`: The map records that currently exist for this concept and project
- `Concept concept`: **NOTE: This is not actually used by the current routines, and should be removed.**
- `MapUser mapUser`: The user requesting the workflow action

and performs some operation upon the set of records as defined by the Workflow Path. The modified set of records is then returned to the Workflow Service.

The `DefaultProjectSpecificAlgorithmHandler` defines these methods for the following workflow paths:

- [Non-legacy Workflow Path](#)
- [Review Project Workflow Path](#)
- [QA Workflow Path](#)
- [Fix Error Workflow Path](#)

Automatically Computation of Mapping Parameters

There are three major elements currently used in automatic computation of mapping parameters and information

- Computing Map Advice
- Computing Map Relations
- Computing target code-specific informational notes

Computing Map Advice

The Mapping Tool will automatically attempt to assign any map advices that should be appended to a map entry when:

- The entry's target code is changed
- A map relation is set

Override the following method to set the logic by which Map Advices are appended:

```
public MapAdviceList computeMapAdvice(MapRecord mapRecord, MapEntry mapEntry)
```

Note that the map entry may not be a managed object (i.e. deserialized from the webapp), which is why the map record this entry belongs to is required.

The `DefaultProjectSpecificAlgorithmHandler` returns an empty list of map advices.

Computing Map Relation

The Mapping Tool will automatically attempt to assign any map advices that should be appended to a map entry when the entry's target code is changed

Override the following method to set the logic by which a Map Relation is appended:

```
public MapRelation computeMapRelation(MapRecord mapRecord, MapEntry mapEntry)
```

Note that the map entry may not be a managed object (i.e. deserialized from the webapp), which is why the map record this entry belongs to is required.

The `DefaultProjectSpecificHandler` returns a null map relation.

Computing Target Terminology Notes

The Mapping Tool will automatically attempt to attach information to the target terminology tree when receiving requests (i.e. searches or expanding a node in the tree) through the Terminology Browser (see [Dashboard Widgets](#)). These notes are contain concept information such as exclusions, inclusions, synonyms, and links to related concepts. Notes are added directly to the persisted `TreePosition` object prior to serialization and returned to the Mapping Tool..

Override the following method if information should be sent to the user when browsing the target terminology:

```
public void computeTargetTerminologyNotes(TreePositionList treePositions)
```

The `DefaultProjectSpecificHandler` does not attach any terminology notes.

Validating Map Records

Validating a Target Code

Target codes are validated at three points in the Tool:

- When a user requests Finish or Publish on a Map Record (see Validating Edited Map Records, below).
- When a release is performed.
- Before a destination terminology tree graph is sent to the Terminology Browser in the user interface.

Override the following method to specify rules for target codes that are eligible for target assignment:

```
public boolean isTargetCodeValid(String terminologyId)
```

The `DefaultProjectSpecificHandler` returns true for all target codes.

Validating Edited Map Records

Map Records are validated prior to any Finish or Publish action requested by a user. The validation consists of two method calls:

- `performUniversalValidationChecks()`
 - Checks that the record has at least one entry.
 - Checks record groups:
 - If the project is not group-based, verifies that only one group, numbered "1", is present.
 - If the project is group-based:
 - verifies that group numbering begins at "1" and is sequential (i.e. no skipped groups)
 - Verifies that high-level map groups do not contain only empty targets
 - Checks entry rules:
 - If the project is not rule-based, verifies no rules are specified
 - If the project is rule-based, verifies that every group is capped with a TRUE rule and that only one TRUE rule exists
 - Checks record for duplicate entries
 - Checks that all advice values are valid for this project

- Checks that all empty targets are properly represented as empty strings instead of null values
- `validateTargetCodes()`
 - For each entry attached to the map record, `isTargetCode valid` is called.
 - If a target code is not valid, an error is appended to the validation result.

Errors are put into a `ValidationResult` object, which contains the following sets of strings

- Messages
- Warnings
- Errors – if this set is not empty, the `ValidationResult`'s method `isValid()` returns false.

If additional checks are required, override the following method:

```
public ValidationResult validateRecord(MapRecord mapRecord)
```

The `DefaultProjectSpecificHandler` calls `performUniversalValidationChecks()` and `validateTargetCodes()`. If this method is overridden, it is recommended that the above checks be included in the overriding code via a "super" call.

Release-specific Validation and Methods

Dependency Modules

At release time, if a dependency modules file should be created, override the following method to determine what is written to the file:

```
public String getModuleDependencyRefSetId()
public Set<String> getDependentModules()
```

The `DefaultProjectSpecificHandler` does not specify any module dependency information

Validating Map Records for Release

If additional checks are required at release time, for example to catch any changed mapping protocols or to verify particular release structure, override the following method:

```
public ValidationResult validateForRelease(ComplexMapRefSetMember member)
```

The `DefaultProjectSpecificHandler` does not perform any additional validation at release time.

Setting Map Relations for Up-propagated records

If additional checks are required at release time, for example to catch any changed mapping protocols or to verify particular release structure, override the following method:

```
public ValidationResult validateForRelease(ComplexMapRefSetMember member)
```

The `DefaultProjectSpecificHandler` does not perform any additional validation at release time.