Extending the Transformation Menu Language

Overview

The menu language may be extended by adding a new location with location-specific features, or by adding a new menu part type.

Adding Locations and Features

To add a location, define a concept extending `TransformationLocation`. Required and optional features are specified using behavior methods:

- Override `getAvailableFeatures()` to return all feature concepts available for the location (both optional and required).
- Override `getRequiredFeatures()` to return all feature concepts required for the location.

To add a feature, define a concept extending `TransformationFeature`.

Menu parts that implement `IExtensibleMenuPart` will have to specify features that are required by the location(s) they are in. Such parts are (generally) generated into a class implementing `interface ActionItem` and additional interfaces as generated by `switch_TransformationLocation_actionItemInterfaces`. The features are generated into class members of the generated menu part class.

You should also extend `switch_TransformationLocation_asStringArray` to specify the identifier(s) that you will use to query items for the location at run time.

Adding Menu Part Types

Extend the `TransformationMenuPart` concept and define the structure, editor, etc. of the custom menu part. A menu part may implement two additional interfaces: `IParameterizableMenuPart`, for parts that support being used inside `TransformationMenuPart_Parameterized`, and `IExtensibleMenuPart`, for parts that have location-specific features or additional parameters for the standard features. For example, `TransformationMenuPart_Action` is both an `IParameterizableMenuPart` and an `IExtensibleMenuPart`, while `TransformationMenuPart_Intention` is only extensible (to provide the wrapped intention as a parameter to its queries) but not parameterizable (since a parameterized intention can be used instead).

Each menu part specified in a section should ultimately be generated into a list of zero or more expressions of type `interface MenuPart` (placeholder parts are generated into zero menu parts).

To generate code for your menu part you need to extend one or two template switches: `switch_TransformationMenuPart_declare` and possibly `switch_TransformationMenuPart_create`.

By default, `switch_TransformationMenuPart_declare` will declare a (static) inner class and attach the `generatedClass` mapping label to it. Next, `switch_TransformationMenuPart_create` generates into a `new` expression creating an instance of the class specified by `generatedClass` mapping label.

If the default behavior fits your use case, it is enough to extend only `switch_TransformationMenuPart_declare` to generate a class implementing `MenuPart`, attaching the `generatedClass` mapping label to it. In case of more complex scenarios you may gain more flexibility by extending both switches.