Create OCL2.0 validation rule

The OCL2.0 validation rule describes the validation logic using OCL2.0.

To create the OCL2.0 validation rule

---

1. Create a constraint.
2. Set the stereotype «UML Standard Profile::Validation Profile::validationRule» for the validation rule.
3. Set the severity level, error message, and abbreviation.
4. Specify constrained element(s).
5. Specify the specification language OCL2.0
6. Enter the OCL2.0 expression as a body of the specification
7. Add/import the created validation rule to a validation suite

The OCL2.0 validation rule can be used in an active validation suite. In this case the validation rule will be executed on any change of constrained elements that are from the validation scope. Executing of the validation rule might be triggered even if no properties important to the validation rule actually changed and this can slow down the active validating speed. In order to avoid degradation of the performance you can create a binary validation rule that uses the OCL2.0 expression to validate model elements but also provides additional information for the modeling tool that allows to optimize triggering of executing after model elements change. For example, we want to check whether all classes have names. This can be done by creating the OCL2.0 validation rule and specifying the OLC2.0 expression:

```
name <> ''
```

The problem is that such validation rule actually is interested in a class property name value change but it will be executed on every property value of a class change. How we can fix this problem and inform the modeling tool to execute the validation rule only on the class property name change:

1. Create a constraint.
2. Set the stereotype «UML Standard Profile::Validation Profile::validationRule» for the validation rule.
3. Set the severity level, error message, and abbreviation.
4. Specify constrained element(s).
5. Specify the specification language OCL2.0
6. Enter the OCL2.0 expression as a body of the specification
7. Create a Java class that extends the *com.nomagic.magicdraw.validation.DefaultValidationRuleImpl* class
8. Override the method *DefaultValidationRuleImpl.getListenerConfigurations()*

```
        public Map<Class<? extends Element>,
Collection<SmartListenerConfig>> getListenerConfigurations()
        {
            Map<Class<? extends Element>, Collection<SmartListenerConfig>>
configs =
                                                new HashMap<Class<?
extends Element>, Collection<SmartListenerConfig>>();
            Collection<SmartListenerConfig> configsForClass = new
ArrayList<SmartListenerConfig>();
            configsForClass.add(SmartListenerConfig.NAME_CONFIG);
            configs.put(com.nomagic.uml2.ext.magicdraw.classes.mdkernel.
Class.class, configsForClass);
            return configs;
        }
```

9. Specify the validation rule's implementation property value - a qualified name of the class
10. Add/import the created validation rule to a validation suite

See the *MyOCLBasedValidationRuleImpl.java* example in the *<program installation directory>\openapi\examples\validation* directory.

The implementation class must be in the classpath of the modeling tool or if the implementation class is in a plugin then the plugin's classloader must be registered to the validation system (see Binary validation rule in the plugin).

Related pages

- Creating validation rules