Creating executable opaque behaviors

Language of an executable opaque behavior can be OCL 2.0, binary, BeanShell, Groovy, JRuby, JavaScript, Jython, or StructuredExpression. Also, it must have the proper number of parameters with proper types.

On this page:

Related Pages

- Model Elements
- Specification Window

How many parameters can an opaque behavior have?

The number of parameters an opaque behavior can have depends on the selected language of the opaque behavior body:

- An OCL 2.0 expression must have a single parameter.
- A binary expression must declare the exact number of parameters has the Java class, to which the expression body of the opaque behavior references.
- Other script expressions, such as JavaScript or Groovy, can have as many parameters as you need.
- Structured expression can have as many parameters as you need.

How to create a parameter for the opaque behavior?

To create a parameter for the opaque behavior

---

1. In the Specification window of the opaque behavior, **Parameters** property group, click the **Create** button. The Specification window of the new parameters opens.
2. Enter a name of the new parameter.
3. Specify the multiplicity of the parameter: either select a value from the drop-down list or type a new one.

   In this case, only the upper bound of the multiplicity is important. For example, the multiplicity [0..1] has the same meaning as the multiplicity [1].

4. Specify the type of the parameter: either select a value from the drop-down list or create a new one by typing directly in the cell.

   Available types:

   - Built-in UML primitives (Integer, Real, String, and so forth)
   - UML element types
   - Java classes (java.io.File, java.util.Properties, and so forth)

5. If the multiplicity upper bound is 1, you may skip this step. Otherwise, specify the following:
   - Whether the arguments are unique.
   - Whether the arguments are ordered.

   If you do not see the Is Unique and Is Ordered properties, select All from the Properties drop-down list in the upper right of the Specification window.

6. Close the Specification window.

How to access the arguments and other values from script body of the opaque behavior?

The following instructions applies to BeanShell, Groovy, JRuby, JavaScript, and Jython scripts only.

To access an argument from a script body, you should refer to the corresponding parameter name.

The script body can access the following values:

- Arguments passed to an opaque behavior as parameters.
- Globally defined values:
  - project (current project)
  - application

How many statements can a script have?

The following instructions applies to BeanShell, Groovy, JRuby, JavaScript, and Jython scripts only.

The script can have multiple statements. In this case the result of the entire script is the result of the last statement.

Limitations in Jython make it so returning a value from the multi-statement Jython script is not straightforward. Instead of:

```
<statement1>
...
<statement n>
<expression returning result>
```

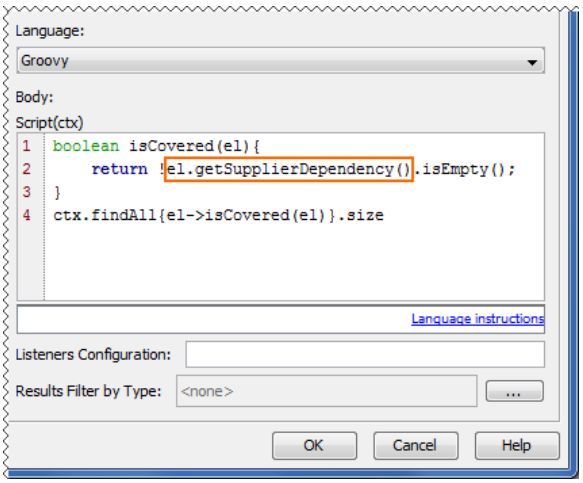
you have to use **result.set(...)**:

```
<statement1>
...
<statement n>
result.set(<expression returning result>)
```

What MagicDraw functionality can a script use?

The following instructions applies to BeanShell, Groovy, JRuby, JavaScript, and Jython scripts only.

The script can call MagicDraw Open API.



More complex model access operations are available in *ModelHelper* and *StereotypesHelper*.

Use import statements to shorten java class names as shown in the following figure.

Language:

Groovy

Body:

filteringPredicate(context)

```
1  import com.nomagic.uml2.ext.jmi.helpers.ModelHelper
2  ModelHelper.isParentOf(scope, context)
```

Short name of java class

Language instructions

Listeners Configuration:

Results Filter by Type: <none> ...

OK Cancel Help