Assign and Declare

<assign> and <declare>

Purpose/Function

Use these tasks to assign values to the environment.

The difference between `declare` and `assign` is essentially the difference between

- {{TYPE **variable** (= *initial value*)}}and
- **`variable`** = *`value`*

That is,

```
int var_n = 0; // DECLARE

var_n = var_q + var_r; // ASSIGN
```

If the assignment or declaration is not global, and assignment is attempted to a non-existent variable, an exception will be thrown; similarly, double declarations on the same frame will cause an exception. As with normal strongly typed languages, however, allowance is made for implicitly repeated declaration inside loops; e.g.:

```
<for ...>
   <declare name="x"/>
   ...
</for>
```

is perfectly acceptable. Global assignment and declaration both automatically overwrite existing values or create the variable if it does not exist.

primitiveAssignables can all be assigned directly as attributes of these tasks. If the attribute is given as an expression, this will be evaluated in the current context prior to assignment.

Otherwise, the task will accept a **single element** to be assigned (the presence of multiple children will cause an exception to be thrown).

See also **Evaluatable Types** for other kinds of values which can be assigned.

Attributes

NAME	TYPE	DEFAULT VALUE	DESCRIPTION
name	`java.lang.String`	required	variable name to declare or assign to
global	`boolean`	`false`	assign to global frame
castTo	`java.lang.String`	`null`	transform number to this numerical type via their string constructors
configured	`boolean`	`true`	(see below for explanation)
boolean	`java.lang.Boolean` or `boolean`	`null`	value to assign
double	`java.lang.Double` or `double`	`null`	value to assign
file	`java.io.File`	`null`	value to assign
long	`java.lang.Long` or `long`	`null`	value to assign
string	`java.lang.String` or `string`	`null`	value to assign
uri	`java.net.URI`	`null`	value to assign
object	`java.lang.Object`	`null`	reference to re-assign to another name

configured:

`true` indicates the child element will be configured in the current context and the resulting instance assigned to the variable name; if `false`, the element's "wrapper" will be placed in the environment, and will be dynamically configured/instantiated in the context of the call each time it is dereferenced. One can similarly delay primitive configuration by assigning a `value` element instead of using the corresponding attribute. If the object to be configured is an evaluatable type, it is both configured and evaluated when assigned; unconfigured evaluatables, however, are configured but not automatically evaluated when dereferenced.

Special Elements

TAG	TYPE	COUNT	DESCRIPTION
<to-string>	java.lang.String	0:1	takes either its child element, or dereferences its TEXT to an object, and calls .toString() on it
<serialized>	java.lang.String	0:1	takes either its child element and serializes it, or dereferences its TEXT to an object and attempts to serialize it (if this object is not *primitiveAssignable* or UserFacing, the task will fail)
<deserialized>	java.lang.Object	0:1	takes its TEXT and attempts to deserialize it using UserFacing deserialization; if the object contains any references to the environment, these will be dereferenced before instantiation (if the object is not UserFacing, the task will fail)
<literal>	java.lang.String	0:1	takes its TEXT and assigns it without dereferencing it in the current context (the value of *configured* is ignored)
<new>	java.lang.Object	0:1	takes its TEXT denoting an absolute tag of an element, or a <java-constructor> element, and constructs an empty new instance of the type. In the case of the constructor, configuration once again must precede instantiation (see below). Can also take a single child element, in which case that element will be fully re-configured each time this task is called (i.e, in a loop).

<new> and <java-constructor>:

When running as a plug-in (inside an OSGI/Eclipse container), it is not possible in general to instantiate arbitrary objects whose type is unknown at compile time using Class.newInstance() due to the OSGI class-loader policies with respect to bundles; hence, only declared extensions can be safely constructed. Otherwise, construction of a fully qualified class name should be possible, but the Java classes of element tags will be unknown unless these have been declared as extensions. Thus:

- IN ECLIPSE: new can always be used, but java-constructor is not guaranteed to work;
- OUTSIDE ECLIPSE: java-constructor should work normally, but new will only work if the tag is a declared extension.

See further Ogrescript Gotchas for uses of <new>.

Examples

- **(0)** null and new Object()

```
<declare taskName="xxx" name="uninitialized"/> <!-- CREATES THE VARIABLE WITH NULL VALUE -->
<declare taskName="yyy" name="anObject">       <!-- CREATES THE VARIABLE WITH new Object VALUE -->
     <new-object/>
</declare>
```

- **(1)** Straight Attribute Values

```
<declare taskName="000" name="v000" string="v000-string-value"/>
<declare taskName="001" name="v001" boolean="false"/>
<declare taskName="002" name="v002" int="2"/>
<declare taskName="003" name="v003" long="3048293841"/>
<declare taskName="004" name="v004" double="3.141592"/>
<declare taskName="005" name="v005" file="/tmp/arossi"/>
<declare taskName="006" name="v006" uri="file:/tmp/arossi"/>
```

- **(2)** Dereferenced Attribute Values

```
<declare taskName="010" name="v010" string="${ra2}"/>
<declare taskName="011" name="v011" boolean="${p1}"/>
<declare taskName="012" name="v012" int="${n1}"/>
<declare taskName="013" name="v013" long="${t2}"/>
<declare taskName="014" name="v014" double="${x0}"/>
<declare taskName="015" name="v015" file="${file0}"/>
<declare taskName="016" name="v016" uri="${uri0}"/>
<declare taskName="017" name="v017" object="${j0}"/>
<declare taskName="018" name="v018" object="${wrapper}"/>
```

- **(3)** Numerical Cast

```
<declare taskName="002" name="v002" long="$E{1000 * 1000}" castTo="int"/>
```

- **(4)** Elements added without configuration / instantiation

```
<declare taskName="020" name="v020" configured="false">
        <value type="long">$E{${t1} - ${t2}}</value>
</declare>

<declare taskName="0201" name="v0201" configured="false">
        <map>
            <map-entry key="a">
                <value type="int">0</value>
            </map-entry>
        </map>
</declare>

<declare taskName="021" name="v021" configured="false">
        <test-task taskName="TestTask021">
                <test-type string="${${a}${a}${a}}"
                        booleanValue="${p0}"         booleanWrapper="${p1}"
                        integerValue="${n0}"         integerWrapper="${n1}"
                        longValue="${t0}"                 longWrapper="${t1}"
                        doubleValue="${x0}"         doubleWrapper="${x1}"
                        value="${wrapper}">
                </test-type>
        </test-task>
</declare>

<declare taskName="022" name="v022" configured="false">
        <map>
                   <map-entry key="queueA">
                           <list>
                                <value>aString</value>
                                <value type="double">3.2</value>
                                <list>
                                        <value>anotherString</value>
                                </list>
                                <list>${l2}</list>
                        </list>
                </map-entry>
                <map-entry key="subMap">
                        <map>${m1}</map>
                </map-entry>
        </map>
</declare>

<declare taskName="023" name="v023" configured="false">
        <trebuchet-pattern base="file:/tmp/arossi">
                <include>*.xml</include>
                <exclude>*.txt</exclude>
        </trebuchet-pattern>
</declare>

<declare taskName="024" name="v024">
        <literal>what does this expression evalutate to? $L{$E{${t1} - ${t2}}} = $E{${t1} - ${t2}}</literal>
</declare>

<declare taskName="025" name="v025" configured="false">
        <serialized>
                <trebuchet-pattern base="${uri0}">
                        <include>${file0}</include>
                </trebuchet-pattern>
        </serialized>
</declare>
```

- **(5)** Elements configured and instantiated before assignment

```
<declare taskName="030" name="v030">
        <value type="long">$E{${t1} - ${t2}}</value>
</declare>

<declare taskName="0301" name="v0301">
        <map>
            <map-entry key="a">
                <value type="int">0</value>
            </map-entry>
        </map>
</declare>

<declare taskName="031" name="v031">
        <test-task taskName="taskWhichShouldntBeHere">
                <test-type string="${${a}${a}${a}}"
                        booleanValue="${p0}"         booleanWrapper="${p1}"
                        integerValue="${n0}"         integerWrapper="${n1}"
                        longValue="${t0}"                longWrapper="${t1}"
                        doubleValue="${x0}"         doubleWrapper="${x1}"
                        value="${wrapper}">
                </test-type>
        </test-task>
</declare>

<declare taskName="032" name="v032">
        <map>
                   <map-entry key="queueA">
                           <list>
                                <value>aString</value>
                                <value type="double">3.2</value>
                                <list>
                                        <value>anotherString</value>
                                </list>
                                <list>${l2}</list>
                        </list>
                </map-entry>
                <map-entry key="subMap">
                        <map>${m1}</map>
                </map-entry>
        </map>
</declare>

<declare taskName="033" name="v033">
        <trebuchet-pattern base="file:/tmp/arossi">
                <include>*.xml</include>
                <exclude>*.txt</exclude>
        </trebuchet-pattern>
</declare>

<declare taskName="sP" name="serializedProperty">
        <serialized>${m1$I{property}}</serialized>
</declare>

<declare taskName="034" name="v034">
        <deserialized>${serializedProperty}</deserialized>
</declare>

<declare taskName="035" name="v035">
        <deserialized>${v025}</deserialized>
</declare>
```

- **(6) <New>**

```
<declare taskName="140" name="v040">
        <new>regex-filter</new>
</declare>

<declare taskName="040" name="v040">
```

```
        <new>
                  <java-constructor className="java.net.URI">
                          <parameter-values>
                                    <value>gridftp://tb1.ncsa.uiuc.edu:2811/home/ncsa/arossi</value>
                          </parameter-values>
                </java-constructor>
        </new>
</declare>

<declare taskName="041" name="v041">
        <new>
                <java-constructor className="java.io.File">
                          <parameter-values>
                                    <ref>${file0}</ref>
                                    <value>subPath</value>
                          </parameter-values>
                </java-constructor>
        </new>
</declare>

<declare taskName="042" name="v042">
        <new>
                <java-constructor className="ncsa.tools.ogrescript.tasks.TestTask">
                           <parameter-values>
                                    <test-type string="${${a}${a}${a}}"
                                      booleanValue="${p0}"         booleanWrapper="${p1}"
                                      integerValue="${n0}"         integerWrapper="${n1}"
                                      longValue="${t0}"                 longWrapper="${t1}"
                                      doubleValue="${x0}"         doubleWrapper="${x1}"
                                      value="${wrapper}">
                                </test-type>
                          </parameter-values>
                </java-constructor>
        </new>
</declare>

<declare taskName="043" name="v043">
        <new>
                <java-constructor className="ncsa.tools.ogrescript.tasks.TestTask">
                            <parameter-values>
                                    <java-constructor className="ncsa.tools.ogrescript.types.TestType"/>
                            </parameter-values>
                </java-constructor>
        </new>
</declare>

<!-- note that one could do the following,
     but the constructor won't be called when dereferenced -->
<declare taskName="044" name="v044">
        <java-constructor className="java.io.File">
                  <parameter-values>
                            <ref>${file0}</ref>
                            <value>subPath</value>
                  </parameter-values>
        </java-constructor>
</declare>


<!-- if this declaration were under an iterator for I, the property
     object would be newly instantiated at each call -->
<declare taskName="045" name="v045">
    <new>
        <property name="i">
           <value>${I}</value>
        </property>
    </new>
</declare>
```

- **(7)** Evaluatables, configured and not configured

```
<declare name="v010" configured="false">
        <expression>${v002} >= ${v004}</expression>
</declare>

<declare name="v011" configured="false">
        <and>
                <is-substring-of string="${v006}" substring="${v005}"/>
                <is-a object="${v004}" fqn="double"/>
        </and>
</declare>

<declare name="v012">
        <date-time type="formatted" timeInMillis="1142540073017"/>
</declare>
```

NOTE

There are special tasks for adding to or assigning to a `List` or `Array` stored in the environment; see the ncsa.tools.ogrescript.tasks.util tasks.