

Gumby Tag Reference

Table of Contents

Gumby API – Tag Library Reference Manual.....	3
General	3
gumby:<Primitive>	3
gumby:<Collection>	4
gumby:Import	4
gumby:Constant	5
gumby:New	6
gumby:Expr	7
gumby:Register	8
gumby:Ref	9
gumby:Resource	10
gumby:URL.....	10
gumby:View	11
Data Binding	12
gumby:BindText	12
gumby:BindCheckBox	13
gumby:BindList	13
gumby:BindCombo	15
Widgets	16
gumby:TablePanel	16
swing:JTable	18

Gumby API – Tag Library Reference Manual

This manual is a work in progress and is provided as a quick reference. Have fun.

General

gumby:<Primitive>

This tag is used to explicitly create instances of Java's primitives. It might be necessary in some contexts, where the framework will assume a `String` if not specified otherwise. The `<Primitive>` placeholder corresponds to one of the following:

- Boolean
- Short
- Int
- Long
- Float
- Double
- String

Attributes:

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
value	A value that should be parseable to the expected primitive.	-	yes

Example

```
...  
<gumby:Int value="100" />  
...
```

gumby:<Collection>

This tag is used to explicitly create instances of `java.util.List`, `java.util.Vector` (many Swing constructors specify vectors as part of their signature), or arrays. The `<Collection>` placeholder corresponds to one of the following:

- Array
- List
- Vector

XML Content:

The tag takes any number of arbitrary children that each should evaluate to an `Object`.

Example

```
...  
<gumby:Array>  
  <gumby:String value="Item 1" />  
  <gumby:String value="Item 1" />  
  <gumby:String value="Item 1" />  
</gumby:Array>  
...
```

gumby:Import

This tag corresponds to Java's import statement. It adds a given package to the import list of the current `RenderContext`'s `Settings`.

Note: the `javax.swing` package is imported by default.

XML Content:

This tag takes a single package name as content. Multiple imports should be used to import multiple packages.

Attributes:

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
name	The name of the constant.	The name consists of the name of the field, and the name of the class that holds that field. If the class has been imported (by default or through configuration), then the package name can be omitted.	yes

Example

```
...  
<gumby:Import>org.acmefoo.mypackage</gumby:Import>  
...
```

gumby:Constant

This tag internally resolves a constant (a public static member of a given class).

Example

```
...  
<swing:JSlider>  
  <orientation>  
    <gumby:Constant name="SwingConstants.HORIZONTAL"/>  
  </orientation>  
  <minorTickSpacing>1</minorTickSpacing>  
  <majorTickSpacing>10</majorTickSpacing>  
  <paintLabels>true</paintLabels>  
  <paintTicks>true</paintTicks>  
  <minimum>0</minimum>  
  <maximum>10</maximum>  
  <value>5</value>  
</swing:JSlider>  
...
```

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
name	The name of the constant.	The name consists of the name of the field, and the name of the class that holds that field. If the class has been imported (by default or through configuration), then the package name can be omitted.	yes

gumby:New

Instantiates a new object. This tag is convenient in cases where a given class does not have a no-args constructor, and therefore cannot be declared as a tag.

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
class	The name of the class from which an object should be instantiated.	The full-qualified name is not necessary if the package of the class has been imported – see <code>gumby:Import</code> .	yes
scopes	The comma-delimited list of scopes that should be searched for the given key/value.	Scopes are searched in the order in which they are specified in the list. Will search all scopes if no scope is specified.	no

Elements

arg

Parent element:

`gumby:New`

Child elements:

value (0-*)

Attributes:

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
type	The full-qualified name of the class of the argument (corresponding to the one in the desired constructor of the class from which to instantiate an object).	Primitive types can be identified as follows: boolean, byte, char, short, int, float, long, double and string.	yes

value

Parent element:

gumby:New

XML Content:

An arbitrary value.

Example

```

...
<gumby:New class="JPanel">
  <arg type="java.awt.LayoutManager">
    <value>
      <gumby:New class="java.awt.GridLayout">
        <arg type="int">
          <value><gumby:Int value="3" /></value>
        </arg>
        <arg type="int">
          <value>
            <gumby:Int value="1" />
          </value>
        </arg>
      </gumby:New>
    </value>
  </arg>
</gumby:New>
...

```

gumby:Expr

Evaluates a given script (which should yield an arbitrary object).

Note that scripts are also provided with a built-in variables:

RenderContext, that corresponds to the

org.sapia.gumby.RenderContext instance in the context of which the script was invoked.

XML Content:

The actual script.

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
lang	bsh (Beanshell), pnuts.	Determines which scripting language is used.	no (defaults to bsh)

Example

```
...
<swing:JButton text="Click once more!!!">
  <actionCommand>click</actionCommand>
  <actionListener>
    <gumby:Expr>
      import javax.swing.JOptionPane;
      import java.awt.event.ActionListener;
      import java.awt.event.ActionEvent;

      new ActionListener() {

        public void actionPerformed(ActionEvent e) {
          frame = RenderContext.getEnv()
            .acquire("Frame", "application");
          JOptionPane.showMessageDialog(frame,
            "Action command: "
            + e.getActionCommand());
        }
      }
    </gumby:Expr>
  </actionListener>
</swing:JButton>
...
```

gumby:Register

Registers an object with a given scope. This that

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
id	A character string of the format id:scope.	The scope part is optional; if omitted, the object is registered with the current context's settings.	yes

Example 1

```
...
<swing:JPasswordField echoChar="*" columns="15">
```

```

    <gumby:Register id="password:forms/UserAccount" />
</swing:JPasswordField>
...

```

Example 2

In the following case, the `gumby:Register` tag will “yield” the **single** tag it encapsulates – having the same outcome as the example above.

```

...
<gumby:Register id="password:forms/UserAccount">
  <swing:JPasswordField echoChar="*" columns="15" />
</gumby:Register>
...

```

gumby:Ref

This tag evaluates to a pre-registered object (see `gumby:Register`).

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
id	A character string of the format <code>id:scope</code> .	The scope part is optional; if not specified, all scopes in the current context will be searched. If no object is found, then the context's settings will be searched. If still nothing is found, this tag yield to null.	yes

Example

```

...
<swing:ButtonGroup>
  <swing:JRadioButton text="choice 1">
    <gumby:Register id="choice1:someForm" />
  </swing:JRadioButton>
  <swing:JRadioButton text="choice 2">
    <gumby:Register id="choice2:someForm" />
  </swing:JRadioButton>
  <swing:JRadioButton text="choice 3">
    <gumby:Register id="choice3:someForm" />
  </swing:JRadioButton>
</swing:ButtonGroup>
<gumby:New class="JPanel">
  <arg type="java.awt.LayoutManager">
    <value>

```

```

    <gumby:New class="java.awt.GridLayout">
      <arg type="int">
        <value><gumby:Int value="3" /></value>
      </arg>
      <arg type="int">
        <value><gumby:Int value="1" /></value>
      </arg>
    </gumby:New>
  </value>
</arg>
<gumby:Ref id="choice1:someForm" />
<gumby:Ref id="choice2:someForm" />
<gumby:Ref id="choice3:someForm" />
</gumby:New>
...

```

gumby:Resource

Yields a `java.io.InputStream` corresponding to a classpath resource.

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
src	The path to the resource.	-	yes

Example

```

...
<gumby:Resource src="some/path/to/resource.xml" />
...

```

gumby:URL

Yields a `java.net.URL` object corresponding to a given character URL – specified as a sting.

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
src	A valid URL string.	-	yes

Example

```
...
<gumby:Resource src="some/path/to/resource.xml" />
...
```

gumby:View

Resolves to a `org.sapia.gumby.view.View` that has been bound to the current context. If no such instance is found under the specified scope, one is internally created.

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
scope	The name of the scope under which the view has been bound.	-	yes

Elements

binding

XML Content:

A single child element that should evaluate to a `org.sapia.gumby.view.Binding` instance.

Example

```
...
<gumby:View scope="forms/UserAccount">
  <binding>
    <gumby:Expr><![CDATA[
      import org.sapia.gumby.view.Binding;
      import org.sapia.gumby.view.View;

      new Binding() {

        public String getId() {
          return "someId";
        }
        public void onBound(View v, Object model) {
          // do something
        }
        public void onUpdated(View v, Object model) {
          // do something
        }
      }
    ]]></gumby:Expr>
  </binding>
</gumby:View>
```

```

        public void onChanged(View v, Object model){
            // do something
        }
        public void updateModel(View v, Object model){
            // do something
        }
    };
}}>
</gumby:Expr>
</binding>
</gumby:View>
...

```

Data Binding

Gumby's databinding tags use Jakarta's JXPath (<http://jakarta.apache.org/commons/jxpath>) to dynamically update models and acquire data from them. Thus, all tags explained below take XPath expressions that will be evaluated following JXPath's conventions.

gumby:BindText

Evaluates to a TextBinding instance that works in conjunction with a `javax.swing.JTextComponent`.

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
id	The id of the JTextComponent in the binding's corresponding View.	-	yes
attribute	a valid XPath expression that will be executed on the model and that should evaluate to the data to bind to the component. Conversely, the expression should match a setter on the model so that the expression will also be used to update the model with the bound data.	-	yes

Example

```
...
<binding>
  <gumby:BindText id="firstName" attribute="firstName" />
</binding>
...
```

gumby:BindCheckBox

Evaluates to a `CheckBoxBinding` instance that works in conjunction with a `javax.swing.JCheckBox`.

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
id	The id of the <code>JCheckBox</code> in the binding's corresponding <code>View</code> .	-	yes
attribute	a valid XPath expression that will be executed on the model and that should evaluate to the data to bind to the component – the data is expected to be of type <code>boolean</code> . Conversely, the expression should match a setter on the model so that the expression will also be used to update the model with the bound data – the setter should of course take a <code>boolean</code> .	-	yes

Example

```
...
<binding>
  <gumby:BindCheckBox id="activated" attribute="activated" />
</binding>
...
```

gumby:BindList

Evaluates to a `ListBinding` instance that works in conjunction with a `javax.swing.JList`.

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
id	The id of the <code>JList</code> in the binding's corresponding <code>View</code> .	-	yes
attribute	a valid XPath expression that will be executed on the model and that evaluates to the <code>java.util.Collection</code> of currently selected data. Conversely, the expression will be used to update the model.	When trying to update the model, the tag implementation will first create a <code>java.util.Collection</code> and attempt to set it on the model. If that fails, the implementation will acquire the <code>Collection</code> corresponding to the XPath expression, call the <code>clear()</code> method on it, and then add the currently selected objects to it.	yes

Elements

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
option	The value should be evaluated to a single arbitrary object that will be added to the <code>JList</code> as one of the available items.	-	no
options	Takes an <code>XML</code> element that should evaluate to a <code>java.util.Collection</code> that holds the available items to be added to the <code>JList</code>	Using this element is convenient if the list of choices is made available "externally" (for example, it could be acquired from an <code>SQL</code> resultset).	no

Example 1.

```
...
<binding>
  <gumby:BindList id="permissions" attribute="permissions">
    <option>Read</option>
    <option>Write</option>
    <option>Delete</option>
  </gumby:BindList>
</binding>
...
```

Example 2.

```
...
<binding>
  <gumby:BindList id="permissions" attribute="permissions">
    <options>
      <gumby:Ref id="myPermissionList:someScope" />
    </options>
  </gumby:BindList>
</binding>
...
```

gumby:BindCombo

Evaluates to a ComboBinding instance that works in conjunction with a javax.swing.JComboBox.

Attributes

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
id	The id of the JComboBox in the binding's corresponding View.	-	yes
attribute	a valid XPath expression that will be executed on the model and that evaluates to the currently selected data. Conversely, the expression will be used to update the model with the item that is selected in the combo.	-	yes

Elements

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
option	The value should be evaluated to a single arbitrary object that will be added to the JComboBox as one of the available items.	-	no
options	Takes an XML element that should evaluate to a java.util.Collection that holds the available items to be added to the JComboBox.	Using this element is convenient if the list of choices is made available "externally" (for example, it could be acquired from an SQL resultset).	no

Example 1.

```
...
<binding>
  <gumby:BindCombo id="language" attribute="language">
    <option>English</option>
    <option>French</option>
    <option>Spanish</option>
    <option>German</option>
  </gumby:BindCombo>
</binding>
...
```

Example 2.

```
...
<binding>
  <gumby:BindList id="role" attribute="role">
    <options>
      <gumby:Ref id="myRoleList:someScope" />
    </options>
  </gumby:BindList>
</binding>
...
```

Widgets

gumby:TablePanel

This tag evaluates to a `org.sapia.gumby.widgets.TablePanel` which is itself an instance of `javax.swing.JPanel`. The tag implementation supports configuration in an HTML-ish way, reminiscent of the `table` tag. It uses `tr` and `td` elements to create cells in the underlying `GridBagLayout`.

Elements

tr

Parent element:

gumby:TablePanel

Child elements:

td (0-*)

Attributes:

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
rowspan	An integer.	Corresponds to GridBagConstraints.gridheight.	no
weight	A double.	Corresponds to GridBagConstraints.weighty.	no

td

Parent element:

tr

XML Content:

An arbitrary value that should evaluate to a `javax.swing.JComponent`.

Attributes:

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
colspan	The full-qualified name of the class of the argument (corresponding to the one in the desired constructor of the class from which to instantiate an object).	Primitive types can be identified as follows: boolean, byte, char, short, int, float, long, double and string.	no
pad	An integer.	Corresponds to GridBagConstraints.ipadx and GridBagConstraints.ipady.	no
padx	An integer.	Corresponds to GridBagConstraints.ipadx.	no
pady	An integer.	Corresponds to GridBagConstraints.ipady.	no
cellpadding	An integer	Corresponds to pad - see above. Provided for HTML-ishness.	no
align	Either left, right or center.		no
valign	Either top, bottom or middle.		no

<i>Name</i>	<i>Value</i>	<i>Comments</i>	<i>Mandatory</i>
weight		Corresponds to GridBagConstraints.weightx.	

Example

```

...
<gumby:TablePanel xmlns:swing="java:swing"
                  xmlns:gumby="gumby:swing">
  <tr>
    <td weight="0.5">
      <swing:JButton>
        <text>Button 1</text>
      </swing:JButton>
    </td>
    <td weight="0.5">
      <swing:JButton>
        <text>Button 2</text>
      </swing:JButton>
    </td>
  </tr>
  <tr>
    <td colspan="2" weight="0.0" pady="40">
      <swing:JButton>
        <text>Long-Named Button 4</text>
      </swing:JButton>
    </td>
  </tr>
</gumby:TablePanel>
...

```

swing:JTable

This tag evaluates to a `javax.swing.JTable`. The tag implementation also supports configuration in an HTML-ish way, using `th`, `tr` and `td` tags.

Elements

th

This tag corresponds to a table header.

Parent element:

swing:JTable

XML Content:

The actual header.

tr

Parent element:

swing:JTable

Child elements:

td (0-*)

td

Parent element:

tr

XML Content:

An arbitrary value that will be the cell's content.

Example

```
...
<swing:JTable>
  <th>HEADER 1</th>
  <th>HEADER 2</th>
  <th>HEADER 3</th>
  <th>HEADER 4</th>
  <tr>
    <td>Cell 1.1</td>
    <td>Cell 1.2</td>
    <td>Cell 1.3</td>
    <td>Cell 1.4</td>
  </tr>
  <tr>
    <td>Cell 2.1</td>
    <td>Cell 2.2</td>
    <td>Cell 2.3</td>
    <td>Cell 2.4</td>
  </tr>
  <tr>
    <td>Cell 3.1</td>
    <td>Cell 3.2</td>
    <td>Cell 3.3</td>
    <td>Cell 3.4</td>
  </tr>
</swing:JTable>
...
```