SwingML

Swing Markup Language Specification

Ezequiel Cuellar
Grati Lozano
Robert Morris

September 2004

INTRODUCTION.

SwingML is a markup language based on XML that is used to render Swing based graphical user interfaces, it provides an alternative to replace completely the use of HTML in the creation of web based applications.
SwingML introduces the concept of a SwingML Renderer. The SwingML Renderer is an applet that is responsible for processing a SwingML form and translating it in to a user interface. The SwingML Renderer is intended to be the front end of a web application.

Note: This document assumes that the reader has good knowledge on Java server side technologies like Servlets, JSP, XML, DTD and Java client side technologies like Swing components and Applets.

CURRENT FEATURES.

SwingML is a project under development. The SwingML Renderer currently supports the following features, components and listeners:

Http POST and GET operations.
Multiple Look & Feel (Metal, Motif, Windows) and third party Look & Feel implementations.
Drag and Drop operations for JList and JTree components.
Custom Event Handlers.
GridLayout, FlowLayout, BorderLayout, GridBagLayout.
JPanel
JTextField
JPasswordField
JLabel
JButton
JTabbedPane
JInternalFrame
JTree
JTable
ButtonGroup
JMenuBar
JMenu
JMenuItem
JList
JSlider
JComboBox
JEditorPane
JDialog
JFrame
JCheckBox
JToolBar
JTextArea
JSplitPane
JOptionPane
JRadioButton
ActionListener
DocumentListener
FocusListener
ItemListener
ListSelectionListener
TreeSelectionListener
TreeExpansionListener
ChangeListener
MouseListener

SOFTWARE REQUIREMENTS.
Java Plug-in 1.4.0_01 or later (Earlier versions are not supported).

This software has been tested with Netscape 7.0, 6.2, Internet Explorer 6.0 and Mozilla FireFox.

SWINGML RENDERER INSTALLATION AND CONFIGURATION.
Download and unzip the distribution file called SwingML.zip. This file contains the following archives:
SwingMLRenderer.jar, SwingML.dtd, SwingML(Uppercase).dtd, src.jar, COPYING and AUTHORS.

The SwingML.dtd is the document type definition that defines the legal building blocks that are used to write a SwingML form. For example if the SwingML form is in a file called "Demo.xml" then this file should make external reference to SwingML.dtd using the following statement:

```
<!DOCTYPE <root element> SYSTEM "<URL to SwingML.dtd>">
```

The SwingMLRenderer.jar contains the bytecode for **the SwingML Renderer and can be run as an application or as an applet.**

To install the SwingML Renderer as an applet use the following tag syntax:

```
<applet
    code="org.swingml.SwingMLRenderer.class"
    archive="<URL pointing to SwingMLRenderer.jar>"
    width="<Desired width>"
    height="<Desired height>"
>
<param
    name="spec"
    value="<URL pointing to SwingML source (Servlet or XML file)>"
>
</applet>
```

The value for the parameter “spec” should be a URL pointing to the server side component that will generate dynamically the SwingML form. The server side component can be a Servlet as shown below

```
<param name=”spec” value=”http://localhost:8080/servlet/DemoServlet”>
```

The Servlet can use XSLT to do a transformation of the data to SwingML or perform a forward to a JSP that will generate dynamically the SwingML form. The Servlet can also generate the SwingML form directly using String concatenation and send it back through the response object. In this case the servlet’s doGet method should look something like this.

```
doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException{
    response.getWriter().print(“---------SwingML form-------”);
}
```

The parameter “spec” can also point directly to an XML file that describes the SwingML form as shown below

```
<param name=”spec” value=”http://localhost8080/Demo.xml”>
```

To run the SwingML Renderer as an application use the following command line statement:

```
java -jar SwingMLRenderer.jar
<URL pointing to SwingML source (Servlet or XML file)> <WIDTH> <HEIGHT>
```

For example:

```
Java –jar SwingMLRenderer.jar file://C:/demo/demo.xml 800 600
```

SERVER SIDE INDEPENDENCE.
It is important to mention that since SwingML is based on XML and HTTP is used to allow the communication between the SwingML Renderer any technology can be used to create the server side component. SwingML is server side independent, all the server side component must care to do is generate the SwingML form and get back the necessary information from the SwingML Renderer out of the POST object.

ASYNCHRONOUS SERVER SIDE CALLS AND GRANULAR UI RENDERING.
Currently when a web browser performs either a GET or POST it has to wait for the server side response before performing other operations. Also the response means repainting the whole UI.
The SwingMLRenderer overcomes these two issues by providing asynchronous server side calls and granular UI rendering. Every GET or POST is performed in a different thread, which means that the UI does not get locked until the server side component responds. Also the response containing the SwingML spec is rendered only in the container where the operation was performed.

SWINGML SPECIFICATION.
The SwingML Specification supports uppercase and lowercase xml elements. By default the SwingML.dtd correspondes to the uppercase spec. A SwingML(Uppercase).dtd is provided as well and must replace SwingML.dtd in case that the uppercase spec is preferred.

DEMO INSTALLATION.
Download the file called Demo.zip and unzip it in C:\, **it is important not to change the destination**.
This will create a folder called C:\Demo. This directory will contain the following files:
Demo.html, Demo.xml, Dialog.xml, GridBagPanel.xml, OptionPane.xml and apply.png.
Copy the SwingMLRenderer.jar and SwingML.dtd files inside C:\Demo and open Demo.html with the web browser.

If nothing comes up is either because the demo files were unzipped in location different than C:\Demo or the Java Plug-in 1.4.0_01 is not properly installed in your web browser.

WRITING A HELLO WORD EXAMPLE.
Note: This example is performed overriding the Demo.xml file. Create a back up of this file and proceed with the instructions.

Clean up the contents of the Demo.xml and start typing the version and the encoding type of this XML document.

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Make external reference to the SwingML document type definition SwingML.dtd

```
<!DOCTYPE PANEL SYSTEM "file://C:/Demo/SwingML.dtd">
```

Just as any Swing based graphical user interface the root component should be always a JPanel so the interface can be dropped inside of a JApplet or a JFrame. A SwingML form follows the same principle, therefore the element that must be always specified first is PANEL.

```
<PANEL NAME="MainPanel">
```

The second component to create is a JLabel containing the text "Hello World!!!". The element used to accomplish this task is LABEL.

```
<LABEL NAME="MainLabel" TEXT="Hello World!!!"/>
```

Finally following the rules of a well-structured XML document the PANEL element should be closed.

```
</PANEL>
```

This is the complete code for the demo:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE PANEL SYSTEM "file://C:/Demo/SwingML.dtd">
<PANEL NAME="MainPanel">
    <LABEL NAME="MainLabel" TEXT="Hello World!!!"/>
</PANEL>
```

Finally save the Demo.xml file and open the Demo.html with the browser.

SPECIFICATION.

PANEL.

```
<PANEL NAME=" " LAYOUT=" " …>
    [SwingML elements]
</PANEL>
```

Attributes: NAME, LAYOUT, BORDER, LAF, EXT-LAF, BEVELTYPE, ORIENTATION, TEXT, ROWS, COLS.

The PANEL element is used to render a JPanel component. A PANEL element must be always the root element of the SwingML form unless the SwingML from is describing a DIALOG or OPTIONPANE elements. The PANEL element can contain any other components including another PANEL. If PANEL is a root element this should be defined in the DOCTYPE statement that is used to specify the location of the SwingML.dtd.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

LAYOUT.
Specifies the layout manager for this element. The allowed values are BorderLayout, GridLayout, FlowLayout and None. If the LAYOUT is omitted then the default value is FlowLayout. If GridLayout is used then it is necessary to include ROWS and COLS. If BorderLayout is used then it is necessary to include ORIENTATION in each one of its children. The value None should be used only when the only child of this PANEL is an INTERNALFRAME. The GridBagLayout is supported by the GRIDBAG-PANEL element.

BORDER.
Specifies the type of border for this element. The allowed values are EtchedBorder and BevelBorder. If the BORDER is omitted then no border will be used to render this component. If BevelBorder is used then it is necessary to specify BEVELTYPE.

LAF (If present it should used only in the root PANEL).
Specifies the Look & Field that will be used to render the whole interface. Its possible values are Metal, Windows and Motif. If the LAF is omitted then the default Look and Field to be used is Metal. This attribute should be used only in the root PANEL.

EXT-LAF (If present it should used only in the root PANEL).

The LAF attribute is provided to support external Look & Feel libraries. This way, a developer can use a Look & Feel other than the three defaults supplied by the JRE (Metal, Windows, & Motif). In order to use the new External Look & Feel, you must provide the following:

1 - An implementation of the swingml.external.ExternalLookAndFeel interface.
2 - Set up your PANEL tag like so:

```
<PANEL LAF="External"
EXT-LAF="com.mycompany.mypackage.MyLookAndFeelImplementation" .../>
```

This will cause the SwingML Renderer to instantiate your ExternalLookAndFeel implementation rather than using the defaults already defined. If the class defined in EXT-LAF does not exist, the panel defaults to "Metal" and writes an exception to the Console.

BEVELTYPE (Required when BORDER is BevelBorder).
Specifies the type of BevelBorder for this element. The allowed values are Raised and Lowered. This attribute can't be omitted if the value of BORDER is BevelBorder.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. Its possible values are North, South, Center, East and West.

TEXT (Required when parent element is TABBEDPANE).
Specifies the text to be used as a displayable name of this element. For example if this element is a child of a TABBEDPANE element then the value of the TEXT attribute will be used as a name of the tab related.

TITLE
Specifies the text to be used as a title for the BORDER of this element. If the BORDER is missing then the TITLE will not be displayed.

ROWS (Required when LAYOUT is GridLayout).
Specifies the number of rows for this element when its LAYOUT is GridLayout.

COLS (Required when LAYOUT is GridLayout).
Specifies the number of columns for this element when its LAYOUT is GridLayout.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

LABEL.

```
<LABEL NAME=”” TEXT=””…/>
```

Attributes: NAME, TEXT, ORIENTATION.
The LABEL element is used to render a JLabel component.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

TEXT (Always required).
Specifies the text that will be displayed by this element.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

ALIGN.
Specifies the alignment of the text in the LABEL element. The allowed values are Left, Center and Right.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

FOREGROUND
Specifies the foreground color of the LABEL element. The allowed values are Black, Blue, Cyan Gray, Green, Magenta, Orange, Pink, Red, White, Yellow.

TEXTFIELD

```
<TEXTFIELD NAME="" TEXT="" …>
    [LISTENER elements]
</TEXTFIELD>
```

Attributes: NAME, TEXT, COLS, EDITABLE, ORIENTATION, TOOLTIP, BACKGROUND, FOREGROUND.

The TEXTFIELD element is used to render a JTextField component. The TEXTFIELD contains LISTENER elements and supports FocusListener and DocumentListener.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.
It specifies the name that will be sent to the server.

TEXT (Always required).
Specifies the initial contents of the TEXTFIELD element. When the form is submitted, the current contents are sent.

COLS (Always required)
Specifies the visible width of the text area.

EDITABLE.
Specifies if the contents of this TEXTFIELD can be modified. The allowed values are True or False.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

ENABLED
Specifies if the TEXTFIELD is enabled. The allowed values are True or False.

BACKBGROUND
Specifies the BACKGROUND color of the TEXTFIELD. The allowed values are: Black, Blue, Cyan, Gray, Green, Magenta, Orange, Pink, Red, White, Yellow.

FOREGROUND
Specifies the FOREGROUND color of the TEXTFIELD. The allowed values are: Black, Blue, Cyan, Gray, Green, Magenta, Orange, Pink, Red, White, Yellow.

BUTTON

```
<BUTTON NAME=”” TEXT=”” …>
      [LISTENER elements]
</BUTTON>
```

Attributes: NAME, TEXT, ORIENTATION, ENABLED, ICON, TOOLTIP.

The BUTTON element is used to render a JButton component. The BUTTON component contains LISTENER elements and supports ActionListener. Differently than HTML the BUTTON element is not sent to the server when the SwingML form is submitted.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

TEXT (Always required)
Specifies the text that will be displayed on this element.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

ENABLED.
Specifies if the BUTTON is enabled.

ICON
Specifies the URL of the image that will be used as an icon for this JButton.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

TABBEDPANE

```
<TABBEDPANE NAME=”” ORIENTATION=””>
      [PANEL and LISTENER elements]
</TABBEDPANE>
```

Attributes: NAME, ORIENTATION, TOOLTIP, CURRENT.

The TABBEDPANE element is used to render a JTabbedPane component. The TABBEDPANE element should contain a PANEL element for every tab desired. The TEXT property value of every child PANEL will be used as the name of the tab related.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.
CURRENT
Specifies the selected index for this TABBEDPANE element.

TREE

```
<TREE NAME=””TEXT=”” …>
    [TREE, NODE and LISTENER elements]
</TREE>
```

Attributes: NAME, TEXT, ORIENTATION, TOOLTIP, DRAGENABLED.

The TREE element is used to render a JTree component. The TREE element contains either TREE elements to display sub trees or NODE elements to display the nodes of a tree or sub tree. LISTENER elements are also supported. When the SwingML form is submitted a SwingML structure representing the current TREE is sent to the server. The server side component is responsible for parsing the structure and extracting its values. The TREE element supports TreeSelectionListener

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.
It specifies the name that will be sent to the server.

TEXT (Always required)
Specifies the text that will be displayed on this element.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

DNDENABLED.
Specifies if the TREE element supports drag and drop operations. The drag and drop support works only locally (save JVM) and TREE and NODE elements can be dragged and dropped in the same TREE element or a different one. The allowed values are True and False.

NODE

<NODE TEXT=""/>

Attributes: TEXT.

The NODE element is used to render a node in a JTree component.

TEXT (Always required)
Specifies the text that will be displayed on this element.

TABLE

```
<TABLE NAME=""ORIENTATION="" …>
    [TC, TR and LISTENER elements]
</TREE>
```

Attributes: NAME, ORIENTATION, TOOLTIP.

The TABLE element is used to render a JTable component. The TABLE element must define first the TC elements for the table columns, then the TR elements for the table rows and finally the LISTENER elements. When the SwingML form is submitted a SwingML structure representing the current TABLE is sent to the server. The server side component is responsible for parsing the structure and extracting its values. The TABLE supports MouseListener.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.
It specifies the name that will be sent to the server.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

POST-STYLE
Allows to specify a different behavior for the POST implementation of the TABLE element. The allowed values are Post-Selected and Post-All.
If the POST-STYLE equals Post-All then all the selected elements of the table will be always sent back to the server side component when performing a POST.
If the POST-STYLE equals Post-Selected or if POST-STYLE is omitted then only the selected elements will be sent to the server side component

POST-COLUMN
Works in conjunction with the POST-STYLE element when its value is Post-Selected. It allows to specify which column in a table (starting with 0) should be sent back to the server side component when performing a POST.

MODE
Specifies the selection mode for this element. The allowed values are Single and Multiple. If missing the default selection mode is Single.

TC

```
<TC TEXT=”” TYPE=” ” …/>
```

Attributes: TEXT, TYPE, ITEMS.

The TC element is used to render a column in a JTable component.

TEXT (Always required)
Specifies the text that will be displayed on this element.

TYPE
Specifies the TYPE of this column. The allowed values are String, Boolean, Combo and Color. If omitted the default value is String.

ITEMS (Required when TYPE is Combo or Color)
Specifies the collection of values to pick from for this column if the TYPE of the TC is Combo or Color. If the TC is of TYPE Combo then ITEMS attribute should contain comma separated alphanumeric values. If the TC is of TYPE Color then the ITEMS attribute should contain comma separated color names. The supported colors are: black, blue, cyan, gray, green, magenta, orange, pink, red, white and yellow. For Boolean TYPE the ITEMS attribute should not be used.

TR

```
<TR>
    [TD elements]
</TR>
```

The TR element is used to render a row in a JTable component. The TR contains TD elements for the values of each of the cells in the table.

TD

```
<TD VALUE=” ” EDITABLE=” ” …/>
```

Attributes: VALUE, EDITABLE, TEXT.

The TD element is used to render the values of the cells in a JTable.

VALUE (Always required)
Specifies the value that will be displayed on this element. The VALUE content should comply with TYPE specified in the TC element or should match one of its ITEMS. If the TYPE of the TC element is Boolean then VALUE should contain ”True” or ”False”.

EDITABLE
Specifies if this table cell can be modified or not. The allowed values are True or False. If omitted the default is False.

TEXT (deprecated)
Specifies the text that will be displayed on this element. This attribute has been deprecated since the inclusion of VALUE.

INTERNALFRAME.

```
<INTERNALFRAME NAME=” ” TEXT=” ” …>
    [SwingML elements]
</INTERNALFRAME>
```

Attributes: NAME, TEXT, LAYOUT, WIDTH, HEIGHT, ROWS, COLS.

The INTERNALFRAME element is used to render a JInternalFrame component. The INTERNALFRAME contains any other SwingML components including another INTERNALFRAME.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.
TEXT
Specifies the text to be used as a displayable title on this element. If this element is omitted then resulting JInternalFrame will lack of title.

LAYOUT.
Specifies the layout manager for this element. The allowed values are BorderLayout, GridLayout, FlowLayout and None. If the LAYOUT is omitted then the default value is FlowLayout. If GridLayout is used then it is necessary to include ROWS and COLS. If BorderLayout is used then it is necessary to include ORIENTATION in each one of its children. The value None should be used only when the only child of this INTERNALFRAME is another INTERNALFRAME. The GridBagLayout is supported by the GRIDBAG-PANEL element.

WIDTH
Specifies the width of this element.

HEIGHT
Specifies the height of this element.
ROWS (Required when LAYOUT is GridLayout).
Specifies the number of rows for this element when its LAYOUT is GridLayout.

COLS (Required when LAYOUT is GridLayout).
Specifies the number of columns for this element when its LAYOUT is GridLayout.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

MENUBAR.

```
<MENUBAR NAME=”” ORIENTATION=””…>
    [MENU elements]
</MENUBAR>
```

Attributes: NAME, ORIENTATION

The MENUBAR element is used to render a JMenuBar component. The MENUBAR element must contain at least one MENU specified and can be used only with PANEL, DIALOG and INTERNALFRAME elements, each one of them must have BorderLayout specified in its LAYOUT property.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

ORIENTATION (Always required).
Specifies the orientation of this element inside its parent. Elements that contain a MENUBAR must specify BorderLayout in its LAYOUT property. The allowed values are North, South, Center, East and West.

MENU.

```
<MENU NAME=”” TEXT=””…>
    [MENU, MENUITEM and SEPARATOR elements]
</MENU>
```

Attributes: NAME, TEXT, MNEMONIC, TOOLTIP.

The MENU element is used to render a JMenu component. The MENU element can contain others components like MENUITEM elements (described below) or MENU itself. It can contain SEPARATORS elements used to display separation lines between different menu items.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

TEXT (Always required).
Specifies the text that will be displayed on this element.

MNEMONIC.
Specifies the keyboard mnemonic or keyboard shortcut on the current model.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains over the component.

MENUITEM.

```
<MENUITEM NAME=”” TEXT=”” …>
    [LISTENER elements]
</MENUITEM>
```

Attributes: NAME, TEXT, MNEMONIC, TOOLTIP, ICON.

The MENUITEM element is used to render a JMenuItem component. The MENUITEM contains LISTENER elements. The MENUITEM supports ActionListener.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

TEXT (Always required)
Specifies the text that will be displayed on this element.

MNEMONIC.
Specifies the keyboard mnemonic or keyboard shortcut on the current model.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

ICON
Specifies the url location of the image that will be displayed in the component.

SEPARATOR

<SEPARATOR NAME=””/>

Attributes: NAME.

The SEPARATOR element is used to display separation lines between different menu items

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

BUTTONGROUP

```
<BUTTONGROUP NAME=” ” TEXT=” ” ....>
    [RADIOBUTTON or CHECKBOX elements]
</BUTTONGROUP>
```

Attributes: NAME, LAYOUT, BORDER, BEVELTYPE, ORIENTATION, TEXT, ROWS, COLS, TOOLTIP.

The BUTTONGROUP element is used to render a JButtonGroup component. The BUTTONGROUP contains RADIOBUTTON or CHECKBOX elements, but it cannot contain both types of elements simultaneously.

NAME (Always required)
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name is not allowed. It can't contain blank spaces and it is always required.

LAYOUT.
Specifies the layout manager for this element. The allowed values are BorderLayout, GridLayout, FlowLayout and None. If the LAYOUT is omitted then the default value is FlowLayout. If GridLayout is used then it is necessary to include ROWS and COLS. If BorderLayout is used then it is necessary to include ORIENTATION in each one of its children.

BORDER.
Specifies the type of border for this element. The allowed values are EtchedBorder and BevelBorder. If the BORDER is omitted then no border will be used to render this component. If BevelBorder is used then it is necessary to specify BEVELTYPE.

BEVELTYPE (Required when BORDER is BevelBorder).
Specifies the type of BevelBorder for this element. The allowed values are Raised and Lowered. This attribute can't be omitted if the value of BORDER is BevelBorder.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. Its possible values are North, South, Center, East and West.

TEXT (Required when parent element is TABBEDPANE).
Specifies the text to be used as a displayable name of this element.

ROWS (Required when LAYOUT is GridLayout).
Specifies the number of rows for this element when its LAYOUT is GridLayout.

COLS (Required when LAYOUT is GridLayout).
Specifies the number of columns for this element when its LAYOUT is GridLayout.
TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

TOOLBAR

```
<TOOLBAR NAME="" ORIENTATION="">
    [BUTTON elements]
</TOOLBAR>
```

Attributes: NAME, ORIENTATION, TOOLTIP.

The TOOLBAR element is used to render a JToolBar component. The TOOLBAR element must contain at least one BUTTON specified and can be used only with PANEL, DIALOG and INTERNALFRAME elements, each one of them must have BorderLayout specified in its LAYOUT property.

NAME (Always required)
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name is not allowed. It can't contain blank spaces and it is always required.

ORIENTATION (Always required).
Specifies the orientation of this element inside its parent. Elements that contain a TOOLBAR must specify BorderLayout in its LAYOUT property. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

SLIDER

```
<SLIDER NAME="" VALUE="" ….>
    [LISTENER elements]
</SLIDER>
```

Attributes: NAME, MINIMUM, MAXIMUM, VALUE, TYPE, ORIENTATION, TOOLTIP, ENABLED.

The SLIDER element is used to render a JSlider component.

NAME (Always required)
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

MINIMUM (Always required)
The MINIMUM attribute is used to specify the minimum numeric range of the SLIDER component.

MAXIMUM (Always required)
The MAXIMUM attribute is used to specify the maximum numeric range of the SLIDER component.

VALUE (Always required)
The VALUE attribute is used to specify the initial numeric value within the range of the SLIDER component.

TYPE (Always required)
The TYPE attribute is used to specify if the SLIDER component should be rendered either in horizontal or vertical position inside its parent. The allowed values are Horizontal and Vertical.

ORIENTATION (Always required)
Specifies the orientation of this element inside its parent. Elements that contain a TOOLBAR must specify BorderLayout in its LAYOUT property. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

ENABLED
Specifies if the SLIDER is enabled. The allowed values are True or False.

EDITORPANE

```
<EDITORPANE NAME=”” PAGE=”” />
```

Attributes: NAME, PAGE.

The EDITORPANE element is used to render a JEditorPane component. The EDITORPANE is intended to display HTML content in a read only mode which is specified by a url. The EDITORPANE element doesn’t accept HTML code embedded within itself in order to keep cleaner the SwingML document.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.

PAIGE (Always required)
The PAIGE attribute specifies the url of the HTML page that will be displayed by the EDITORPANE element.

EDITABLE
Specifies if the content of the EDITORPANE can be modified. The allowed values are True or False.

COMBOBOX

```
<COMBOBOX  NAME=”” ORIENTATION=””>
      [ITEM elements and LISTENER elements]
</COMBOBOX>
```

Attributes: NAME, ORIENTATION, TOOLTIP.

The COMBOBOX element is used to render a JComboBox component. The COMBOBOX component contains ITEM elements to render JComboBox items and LISTENER elements. The COMBOBOX supports ItemListener. In a COMBOBOX element only one ITEM can be selected.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.
It specifies the name that will be sent to the server.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

EDITABLE
Specifies if the COMBOBOX element is editable. An editable COMBOBOX element allows the user to type into the field or select an item from the list to initialize the field, after which it can be edited.
The allowed values are True or False.

LIST

```
<LIST  NAME=”” ORIENTATION=””>
    [ITEM elements and LISTENER elements]
</LIST>
```

Attributes: NAME, MODE, ORIENTATION, TOOLTIP, DRAGENABLED.
The LIST element is used to render a JList component. The LIST element contains ITEM elements to render JList items and LISTENER elements. The LIST element supports ListSelectionListener. In a LIST element multiple items can be selected. When the SwingML form is submitted the selected values of the list are sent separated by a comma.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name are not allowed. It can't contain blank spaces and it is always required.
It specifies the name that will be sent to the server.

MODE
Specifies the selection mode for this element. The allowed values are Single and Multiple. If missing the default selection mode is Single.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

POST-STYLE
Allows to specify a different behavior for the POST implementation of the LIST element. The allowed values are Post-Selected and Post-All.
If the POST-STYLE equals Post-All then all the ITEM elements of the list will be always sent back to the server side component when performing a POST.
If the POST-STYLE equals Post-Selected or if POST-STYLE is omitted then only the selected ITEM elements will be sent to the server side component

DNDENABLED.
Specifies if the LIST element supports drag and drop operations. The drag and drop support works only locally (save JVM) and ITEM elements can be dragged and dropped in the same LIST element or a different one. The allowed values are True and False.

ITEM

<ITEM TEXT=”” SELECTED=””/>

Attributes: TEXT, SELECTED.

The ITEM element is used to render items in a JList and JComboBox components.

VALUE
Specifies the value that will be displayed on this element.

TEXT (Deprecated)
Specifies the text that will be displayed on this element.

SELECTED
Specifies that the item is selected in the element. The allowed values are True and False.
If omitted then the item won’t be selected in the element. For a JList multiple items can have SELECTED equals True, in a JComboBox component only one item can have SELECTED equals True.

DIALOG

```
< DIALOG NAME=” ” HEIGHT =” ” WIDTH=” “ ...>
        [SwingML elements]
</DIALOG>
```

Attributes: NAME, WIDTH, HEIGHT, LAYOUT, TEXT, ROWS COLS, MODAL

The DIALOG element is used to render a JDialog component. The DIALOG element should be described in a separate SwingML form and must be used as a root element in it. Rendering a DIALOG element is described in the section “Rendering a FRAME, DIALOG or an OPTIONPANE” below. The DIALOG component contains other SwingML elements. If DIALOG is a root element then this should be defined in the DOCTYPE statement that is used to specify the location of the SwingML.dtd.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name is not allowed. It can't contain blank spaces and it is always required.

WIDTH (Always required).
Specifies the width of this element.

HEIGHT (Always required).
Specifies the height of this element.

LAYOUT
Specifies the layout manager for this element. The allowed values are BorderLayout, FlowLayout, GridLayout and None. If the LAYOUT is omitted then the default value is FlowLayout. If GridLayout is used then it is necessary to include ROWS and COLS. If BorderLayout is used then it is necessary to include ORIENTATION in each one of its children. The GridBagLayout is supported by the GRIDBAG-PANEL element.

TEXT
Specifies the text to be used as a displayable title on this element. If this element is omitted then resulting JDialogBox will lack of title.

ROWS (Required when LAYOUT is GridLayout).
Specifies the number of rows for this element when its LAYOUT is GridLayout.
COLS (Required when LAYOUT is GridLayout).
Specifies the number of columns for this element when its LAYOUT is GridLayout.

MODAL
Specifies if this JDialog is modal. The allowed values are True and False.

FRAME

```
< FRAME NAME=" " HEIGHT =" " WIDTH=" " ...>
            [SwingML elements]
</FRAME>
```

Attributes: NAME, WIDTH, HEIGHT, LAYOUT, TEXT, ROWS COLS

The FRAME element is used to render a JFrame component. The FRAME element should be described in a separate SwingML form and must be used as a root element in it. Rendering a FRAME element is described in the section "Rendering a FRAME, DIALOG or an OPTIONPANE" below. The FRAME component contains other SwingML elements. If FRAME is a root element then this should be defined in the DOCTYPE statement that is used to specify the location of the SwingML.dtd.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name is not allowed. It can't contain blank spaces and it is always required.

WIDTH (Always required).
Specifies the width of this element.

HEIGHT (Always required).
Specifies the height of this element.

LAYOUT
Specifies the layout manager for this element. The allowed values are BorderLayout, FlowLayout, GridLayout and None. If the LAYOUT is omitted then the default value is FlowLayout. If GridLayout is used then it is necessary to include ROWS and COLS. If BorderLayout is used then it is necessary to include ORIENTATION in each one of its children. The GridBagLayout is supported by the GRIDBAG-PANEL element.

TEXT
Specifies the text to be used as a displayable title on this element. If this element is omitted then resulting JDialogBox will lack of title.

ROWS (Required when LAYOUT is GridLayout).
Specifies the number of rows for this element when its LAYOUT is GridLayout.

COLS (Required when LAYOUT is GridLayout).
Specifies the number of columns for this element when its LAYOUT is GridLayout.

CHECKBOX

```
<CHECKBOX NAME=" " TEXT=" " ....>
    [LISTENER elements]
</CHECKBOX>
```

Attributes: NAME, TEXT, CHECKED, ORIENTATION, TOOLTIP

The CHECKBOX element is used to render a JCheckBox component. The CHECKBOX contains LISTENER elements and supports ItemListener.itemStateChanged.selected and ItemListener.itemStateChanged.deselected.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name is not allowed. It can't contain blank spaces and it is always required.
TEXT (Always required)
Specifies the text that will be displayed on this element.

CHECKED
Specifies the element's state. The values allowed are 'True' and 'False'. If CHECKED is omitted the default value is 'False'.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

ENABLED.
Specifies if the CHECKBOX is enabled. The allowed values are True or False.

TEXTAREA

```
<TEXTAREA NAME="" COLS="" ROWS="" ...>
    [TEXT elements and LISTENER elements]
</TEXTAREA>
```

Attributes: NAME, COLS, ROWS, ORIENTATION, TOOLTIP.

The TEXTAREA element is used to render a JTextArea component. The TEXTAREA contains TEXT and LISTENER elements and supports supports DocumentListener and FocusListener.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name is not allowed. It can't contain blank spaces and it is always required.

COLS(Always required)
Specifies the visible width for this element.

ROWS (Always required)
Specifies the number of visible rows for this element.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

LINE-WRAP
Specifies if the TEXTAREA element will do line wrapping. The allowed values are True or False.

FOREGROUND
Specifies the foreground color of the TEXTAREA element. The allowed values are Black, Blue, Cyan Gray, Green, Magenta, Orange, Pink, Red, White, Yellow.

BACKGROUND
Specifies the foreground color of the TEXTAREA element. The allowed values are Black, Blue, Cyan Gray, Green, Magenta, Orange, Pink, Red, White, Yellow.

SPLITPANE

```
<SPLITPANE NAME=" " SIZE =" " LOCATION=" " ....>
    [PANEL elements and GRIDBAG-PANEL elements]
</SPLITPANE>
```

Attributes: NAME, SIZE, LOCATION, TYPE, ORIENTATION, TOOLTIP.

The SPLITPANE element is used to render a JSplitPane component. The SPLITPANE element can contain one or more PANEL or GRIDBAG-PANEL element.

NAME (Always required)
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name is not allowed. It can't contain blank spaces and it is always required.

SIZE (Always required)
Specifies the size of the divider

LOCATION (Always required)
Specifies either the x or y position of the divider, depending on the orientation of the JSplitPane
TYPE
Specifies how the views are split. Allowed values are Vertical and Horizontal.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

SCROLLBARS
Specifies if the SPLITPANE element uses scrollbars to display its elements. The allowed values are True or False.

OPTIONPANE

```
<OPTIONPANE TEXT=" " TYPE=" ">
      [LISTENER elements]
</OPTIONPANE>
```

Attributes: TEXT, TYPE.

The OPTIONPANE element is used to render a JOptionPane dialog box. The OPTIONPANE element should be described in a separate SwingML form and must be used as a root element in it. If OPTIONPANE is a root element this should be defined in the DOCTYPE statement that is used to specify the location of the SwingML.dtd. Rendering a OPTIONPANE element is described in the section "Rendering a FRAME, DIALOG or an OPTIONPANE" below.
The OPTIONPANE contains LISTENER elements and supports ActionListener.actionPerformed.yes, ActionListener.actionPerformed.no and ActionListener.actionPerformed.cancel.

TEXT (Always required)
Specifies the descriptive message to be placed in the dialog box.

TYPE (Always required)
Specifies the type of dialog box displayed. Allowed values are Confirm, Input and Message.

RADIOBUTTON

```
<RADIOBUTTON NAME=" " TEXT=" " ....>
      [LISTENER elements]
</RADIOBUTTON>
```

Attributes: NAME, TEXT, CHECKED, ORIENTATION, TOOLTIP.

The RADIOBUTTON element is used to render a JRadioButton component. The RADIOBUTTON contains LISTENER elements and supports ItemListener.itemStateChanged.selected and ItemListener.itemStateChanged.deselected

NAME (Always required)
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name is not allowed. It can't contain blank spaces and it is always required.

TEXT (Always required)
Specifies the text that will be displayed on this element

CHECKED
Specifies the state of the component. Allowed values are 'True' and 'False'. If CHECKED is omitted the default value is 'False'.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. The allowed values are North, South, Center, East and West.

TOOLTIP
Specifies the comment that will be displayed when the mouse remains a piece of time over the component.

LISTENER

```
<LISTENER EVENT="">
    [ACTION elements and EXTERNAL-ACTION elements]
</LISTENER>
```

Attributes: EVENT.

The LISTENER element is used to register listeners in any SwingML element. The LISTENER contains several ACTION or EXTERNAL-ACTION elements.

EVENT (Always required)
Specifies the event that this LISTENER is registered for. A SwingML element can't have two listeners for the same EVENT assigned to the same component. The allowed values depending of the supported listeners by the SwingML element are:

ActionListener.actionPerformed
ActionListener.actionPerformed.yes
ActionListener.actionPerformed.no
ActionListener.actionPerformed.cancel
DocumentListener.insertUpdate
DocumentListener.changedUpdate
DocumentListener.removeUpdate
FocusListener.focusGained
FocusListener.focusLost
ListSelectionListener.valueChanged
TreeSelectionListener.valueChanged
ItemListener.itemStateChanged
ItemListener.itemStateChanged.selected
ItemListener.itemStateChanged.deselected
MouseListener.doubleClicked
MouseListener.singleClicked
TabListener.stateChanged
TreeExpansionListener.treeCollapsed
TreeExpansionListener.treeExpanded
TreeExpansionListener.treeWillCollapse
TreeExpansionListener.treeWillExpand

ACTION

<ACTION COMPONENT="" METHOD="" …/>
Attributes: COMPONENT, METHOD, TYPES, VALUES.

The ACTION element defines the action to take when the EVENT for the LISTENER is executed.

COMPONENT (Always required)
Specifies the NAME of the component that will be affected by this action. The NAME must be the name of a SwingML element previously defined in this form.

METHOD (Always required)
Specifies the METHOD to call on the COMPONENT when this ACTION is executed.

TYPES
Specifies the TYPES of the VALUES that the METHOD of the COMPONENT affected by this ACTION takes. If the method takes more than one type then the entered TYPES must be comma separated. The possible values for the TYPES attribute can be any primitive (int, byte,…) or wrapper (String, Double…).

VALUES
Specifies the VALUES for the METHOD of the COMPONENT affected by this action. If the method takes more that one value then the entered VALUES must be comma separated.

EXTERNAL-ACTION

```
<EXTERNAL-ACTION COMPONENT=”” EXTERNAL-CLASS=””>
    [ACTION-PARAM elements]
</EXTERNAL-ACTION>
```

Attributes: COMPONENT, EXTERNAL-CLASS.

The EXTERNAL-ACTION element conforms the mechanism used to support custom event handlers. For more information about how to implement custom event handlers please refer to the document “Custom Event Handlers”. The EXTERNAL-ACTION element defines the class that contains the custom behavior that will be performed when the EVENT for the LISTENER is executed

COMPONENT (Always required)
Specifies the NAME of the component that will be affected by this action. The NAME must be the name of a SwingML element previously defined in this form.

EXTERNAL-CLASS (Always required)
Specifies the name of the class that implements the custom event handler.

ACTION-PARAM

```
<ACTION-PARAM NAME=”” VALUE=”” />
```

Attributes: NAME, VALUE.

The ACTION-PARAM element is used to pass parameters to the class that implements the custom event handler. These parameters are used to obtain references to any component that have been rendered.

NAME (Always required)
Specifies the name of the parameter.

VALUE (Always required)
Specifies the NAME of the component that will be referenced in the class that implements the custom event handler. The NAME must be the name of a SwingML element previously defined in this form.

GRIDBAG-PANEL

```
<GRIDBAG-PANEL NAME=”” TEXT=”” …>
    [GRIDBAG-ROW elements]
</GRIDBAG-PANEL>
```

Attributes: NAME, BORDER, BEVELTYPE, ORIENTATION, TEXT, PADDING, IPADX, IPADY, FILL, ANCHOR, WEIGHTX, WEIGHTY, GRIDHEIGHT, GRIDWIDTH.

The GRIDBAG-PANEL element is used to render a JPanel whose layout is GridBagLayout. The GRIDBAG-PANEL can contain one or more GRIDBAG-ROW elements.

NAME (Always required).
The NAME attribute is the key that uniquely identifies this element across the form therefore two elements with the same name is not allowed. It can't contain blank spaces and it is always required.

BORDER.
Specifies the type of border for this element. The allowed values are EtchedBorder and BevelBorder. If the BORDER is omitted then no border will be used to render this component. If BevelBorder is used then it is necessary to specify BEVELTYPE.

BEVELTYPE (Required when BORDER is BevelBorder).
Specifies the type of BevelBorder for this element. The allowed values are Raised and Lowered. This attribute can't be omitted if the value of BORDER is BevelBorder.

ORIENTATION (Required when LAYOUT is BorderLayout).
Specifies the orientation of this element inside its parent. This attribute can only be used if its parent LAYOUT is BorderLayout. Its possible values are North, South, Center, East and West.

TEXT (Required when parent element is TABBEDPANE).
Specifies the text to be used as a displayable name of this element. For example if this element is a child of a TABBEDPANE element then the value of the TEXT attribute will be used as a name of the tab related.

PADDING
Specifies the padding or insets around the component.

IPADX, IPADY
These fields specify the internal padding to add to the component’s minimum size. The new size will be the old size plus twice the padding

FILL
The size of a row or a column is determined by the widest/tallest element contained in them. FILL specifies what to do to an element that is smaller that this size. The allowed values are None, Horizontal, Vertical and Both. None means not to expand the element at all, Horizontal means to expand horizontally but not vertically, Vertical means to expand vertically but not horizontally and Both means to expand the element horizontally and vertically.

ANCHOR
If FILL is set to None then ANCHOR determines where the element gets placed.
The allowed values are Center, North, South, East, West, NorthWest, NorthEast, SouthEast and SouthWest. Center is used to center the component and North, South, East, West, NorthWest, NorthEast, SouthEast and SouthWest are used to position one side or corner the allocated box.

WEIGHTX, WEIGHTY
These fields determine how much the component will stretch in the x or y direction if space is left over after sizing each column based on the widest object and each row based on the tallest. A % sign must precede the values.

GRIDHEIGHT, GRIDWIDTH
These fields determine the number of columns and rows the component occupies.

GRIDBAG-ROW

```
<GRIDBAG-ROW PADDING="" FILL="" …>
    [GRIDBAG-CELL elements]
</GRIDBAG-ROW>
```

Attributes: NAME, PADDING, IPADX, IPADY, FILL, ANCHOR, WEIGHTX, WEIGHTY, GRIDHEIGHT, GRIDWIDTH,.

The GRIDBAG-ROW element is used to define the rows of a GRIDBAG-PANEL element. The GRIDBAG-ROW element can contain one or more GRIDBAG-CELL elements.

NAME
Specifies the name of the GRIDBAG-ROW

PADDING
Specifies the padding or insets around the component.

IPADX, IPADY
These fields specify the internal padding to add to the component's minimum size. The new size will be the old size plus twice the padding

FILL
The size of a row or a column is determined by the widest/tallest element contained in them. FILL specifies what to do to an element that is smaller that this size. The allowed values are None, Horizontal, Vertical and Both. None means not to expand the element at all, Horizontal means to expand horizontally but not vertically, Vertical means to expand vertically but not horizontally and Both means to expand the element horizontally and vertically.

ANCHOR
If FILL is set to None then ANCHOR determines where the element gets placed.
The allowed values are Center, North, South, East, West, NorthWest, NorthEast, SouthEast and SouthWest. Center is used to center the component and North, South, East, West, NorthWest, NorthEast, SouthEast and SouthWest are used to position one side or corner the allocated box.

WEIGHTX, WEIGHTY
These fields determine how much the component will stretch in the x or y direction if space is left over after sizing each column based on the widest object and each row based on the tallest. A % sign must precede the values.

GRIDHEIGHT, GRIDWIDTH
These fields determine the number of columns and rows the component occupies.

GRIDBAG-CELL

```
<GRIDBAG-CELL PADDING=”” FILL=”” …>
    [SwingML elements]
</GRIDBAG-CELL>
```

Attributes: NAME, PADDING, IPADX, IPADY, FILL, ANCHOR, WEIGHTX, WEIGHTY, GRIDHEIGHT,GRIDWIDTH.

The GRIDBAG-CELL element is used to define the cells of a GRIDBAG-ROW element. The GRIDBAG-CELL contains any other components including another GRIDBAG-PANEL.
NAME
Specifies the name of the GRIDBAG-CELL
PADDING
Specifies the padding or insets around the component.

IPADX, IPADY
These fields specify the internal padding to add to the component’s minimum size. The new size will be the old size plus twice the padding

FILL
The size of a row or a column is determined by the widest/tallest element contained in them. FILL specifies what to do to an element that is smaller that this size. The allowed values are None, Horizontal, Vertical and Both. None means not to expand the element at all, Horizontal means to expand horizontally but not vertically, Vertical means to expand vertically but not horizontally and Both means to expand the element horizontally and vertically.

ANCHOR
If FILL is set to None then ANCHOR determines where the element gets placed.
The allowed values are Center, North, South, East, West, NorthWest, NorthEast, SouthEast and SouthWest. Center is used to center the component and North, South, East, West, NorthWest, NorthEast, SouthEast and SouthWest are used to position one side or corner the allocated box.

WEIGHTX, WEIGHTY
These fields determine how much the component will stretch in the x or y direction if space is left over after sizing each column based on the widest object and each row based on the tallest. A % sign must precede the values.

GRIDHEIGHT, GRIDWIDTH
These fields determine the number of columns and rows the component occupies.

DATA-STORE

```
<DATA-STORE>[Any]</DATA-STORE>
```

The DATA-STORE element is a little more than a way to store text information. It can be used to define functions and then include them in script invocations or it can be used to store any kind of data that does not find its way onto the screen.

DEBUG

<DEBUG>[TEXT elements]</DEBUG>

The DEBUG element is used to define debug messages. The DEBUG element contains TEXT elements.

TEXT

<TEXT>[Debug message]</TEXT>

The TEXT element contains the debug message that will be posted to the java console.

SUBMITTING THE FORM.
Submitting the form is done through an ACTION. Any SwingML element that currently supports LISTENER elements can perform a submit.
Performing a submit is done calling the special method “submit” that is implemented in the PANEL DIALOG or INTERNALFRAME element. For example lets say that we have a BUTTON element that will perform the submit, in this case we have to create a LISTENER that has an ACTION element that calls this method. The following code shows this.

```
<BUTTON NAME="TestButton" TEXT="Accept">
  <LISTENER EVENT="ActionListener.actionPerformed">
    <ACTION
       COMPONENT=”<NAME of the parent element>"
       METHOD="submit" TYPES="String"
       VALUES="<URL of the server side component>"/>
  </LISTENER>
</BUTTON>
```

Warning: Only the children of the specified COMPONENT are included in the POST operation.
It is important that the COMPONENT attribute specifies the correct parent element that can be a PANEL, DIALOG or INTERNALFRAME element. The reason is that an XML and value parsing translation is done by performing an iteration starting from the parent component in the rendered interface through all the children in it. This will ensure that only the values of all children components for the specified parent will be included in the POST operation. If the incorrect parent is specified then the expected values will not be included. Once the POST is completed the response from the server side component containing the SwingML spec will be rendered only in the component that performed the operation. Every server side call is asynchronous meaning that the SwingMLRenderer will not lock until the operation is processed.

RENDERING A FRAME, DIALOG OR AN OPTIONPANE.
Rendering a DIALOG or an OPTIONPANE is done through an ACTION. Any SwingML element that currently supports LISTENER elements can render any of these elements.
Performing a rendering is done calling the special method “render” that is implemented in the PANEL DIALOG or INTERNALFRAME element. For example lets say that we have a BUTTON element that will render a DIALOG, in this case we have to create a LISTENER that has an ACTION element that calls this method. The following code shows this.

```
<BUTTON NAME="TestButton" TEXT="Accept">
  <LISTENER EVENT="ActionListener.actionPerformed">
    <ACTION
       COMPONENT=”<NAME of the root element>"
       METHOD="render" TYPES="String, String"
       VALUES="<URL of the server side component>, <NAME of the parent element>"/>
  </LISTENER>
</BUTTON>
```

It is important that the COMPONENT attribute takes the name of the root element that can be a PANEL, DIALOG or INTERNALFRAME element. The value of VALUES must be the URL to the server side component that will generate dynamically the SwingML form that describes the DIALOG and the NAME of the element that will be acting as a parent. The server side component can be either an XML file or a Servlet. In the last case the Servlet can directly create the SwingML form using String concatenation, it can use XSLT to transform the data to SwingML or it can forward to a JSP to create the SwingML form. Every server side call is asynchronous meaning that the SwingMLRenderer will not lock until the operation is processed

OPENING A SECOND BROWSER WINDOW.
Opening a second web browser window is done is done calling the special method "showDocument". It takes two parameters, which are the URL of the server side component, and the target argument, which indicates where to display the HTML page.
The Target argument is interpreted as follows:

"_self"	Show in the window and frame that contain the applet.
"_parent"	Show in the applet's parent frame. If the applet's frame has no parent frame, acts the same as "_self".
"_top"	Show in the top-level frame of the applet's window. If the applet's frame is the top-level frame, acts the same as "_self".
"_blank"	Show in a new, unnamed top-level window.
"name"	Show in the frame or window named name. If a target named name does not already exist, a new top-level window with the specified name is created, and the document is shown there.

```
<BUTTON NAME="TestButton" TEXT="Accept">
  <LISTENER EVENT="ActionListener.actionPerformed">
    <ACTION
      COMPONENT="<NAME of the parent element>"
      METHOD="showDocument" TYPES="String, String"
      VALUES="<URL of the server side component>, <Target argument>"/>
  </LISTENER>
</BUTTON>
```