SwingML Tutorial

Last Modified: 7/10/2007 12:22:37 PM

Introduction

A SwingML user interface is created using XML tags. Similar to HTML tags, SwingML tags exist that define SwingUI component attributes and events. The XML tags are read by the SwingML renderer class, and a Swing user interface is rendered. ***(Note: SwingML elements and attributes must be defined in upper case.)***

This document describes how to construct a simple SwingML user interface. It is purposely simplistic, so that the features of SwingML can be understood. Each section of this document introduces user interface concepts and features by applying them to a common example.

Setup

Make sure the **SwingML<version>.jar** and supporting jar files are in the JRE class path. Refer to the **readme.txt** file included with the download file a list of required jar files.

Execution Environment

The **SwingMLRenderer** classes **main(…)** method is invoked and supplied with arguments that define a URL referencing a file that contains XML, and width and height values. An example is shown below.

java org.swingml.SwingMLRenderer <url> <width> <height>

You can define an XML in a locale file directory and render with a file URL, or for a more dynamic approach you can create a JSP with XML tags to an application server and specify XML that way.

XML tags defined in a File URL

file:///drive:|<path>/example.xml

XML tags defined in a JSP

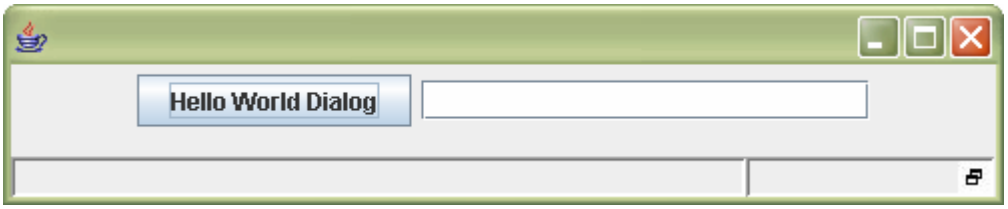
http://<host:port>/example.jsp

SwingML DTD

XML files are validated using the supplied SwingML DTD. Make sure this file is located on the path indicated by the **DOCTYPE** directive, if one is specified in the XML document.

Simple as it gets Hello World User Interface…

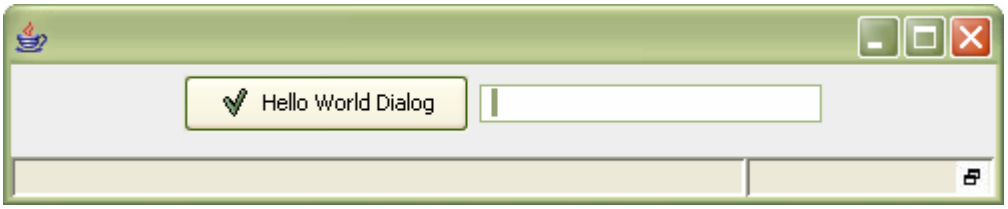
A simple hello world user interface Swing interface and the SwingML tags that define it are shown below.




```
<!DOCTYPE PANEL SYSTEM "WebContent/SwingML.dtd">
<PANEL NAME="helloPanel" >
     <BUTTON NAME="helloButton" TEXT="Hello World">

    </BUTTON>

    <TEXTFIELD NAME="helloField" COLS=”20”>

    </TEXTFIELD>
</PANEL>
```

Defining Attributes

Component attributes are defined as XML element attributes. For example, assigning an icon to a button, or defining look and feel for a Panel are done with attributes. The HelloWorld UI screen and XML definitions below applies these attributes,




```
<!DOCTYPE PANEL SYSTEM "WebContent/SwingML.dtd">
<PANEL NAME="helloPanel" LAF="Windows">

     <BUTTON NAME="helloButton" TEXT="Hello World" ICON="apply.png">

    </BUTTON>
```

```
    <TEXTFIELD NAME="HelloField" COLS=”20”>

    </TEXTFIELD>

</PANEL>
```

Layout Managers

Swing component positions and relationships to one another are based upon an assigned layout manager. Layout managers are assigned to components (Panels, Dialogs, etc…) that contain other components. By default the Flow layout is applied, this can be overridden by setting the LAYOUT Manager attribute.

The GRID layout manager is applied to the case study to accommodate one row of elements and two columns. The resulting screen shot and XML is shown below.


```
<!DOCTYPE PANEL SYSTEM "WebContent/SwingML.dtd">
<PANEL NAME="helloPanel" LAF="Windows" LAYOUT="GridLayout" ROWS="1"
COLS="2">

     <BUTTON NAME="helloButton" TEXT="Hello World" ICON="apply.png">

    </BUTTON>

    <TEXTFIELD NAME="helloField" COLS=”20”>

    </TEXTFIELD>

</PANEL>
```

Event Handling

Events for a given component are handled by listening for events broadcasted by a component. When an event occurs, an action defined for the event listener is performed.

Listeners and actions are assigned to components by nesting XML tags inside a component XML definition. Adding action performed listeners to the two buttons in the example user interface is shown below.

```
<BUTTON NAME="helloButton" TEXT="Hello World" ICON="apply.png">
      <LISTENER EVENT="ActionListener.actionPerformed">

      </LISTENER>
```

```
</BUTTON>

<TEXTFIELD NAME="helloField" COLS="20" >
  <LISTENER EVENT="KeyListener.keyReleased">
    <ACTION COMPONENT="helloField" METHOD="setText" TYPES="String"
VALUES="Hello World"/>
 </LISTENER>
</TEXTFIELD>
```

Client Side Actions

Listener actions are defined to execute public methods against a component instance defined in the user interface XML. An example of a local method invocation is shown in the action defined for the <TEXTFIELD> component. When a keyReleased event occurs inside the text field, the text of the field is set to "HelloWorld".

```
<TEXTFIELD NAME="helloField" COLS="20" >
  <LISTENER EVENT="KeyListener.keyReleased">
   <ACTION COMPONENT="helloField" METHOD="setText" TYPES="String"
VALUES="Hello World"/>
 </LISTENER>
</TEXTFIELD>
```

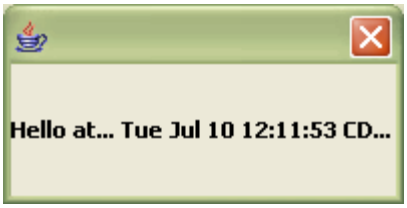
Server Side Actions

A more useful way of reacting to component events is to use the <CONTROLLER-ACTION> element that provides a more J2EE style of programming. This element allows a URL to be defined which could be some kind of server side resource, such as a JSP, Servlet, or Struts action that can process HTTP request parameters and return XML as a result. This action definition also has an operation attribute that indicates whether the resulting XML will refresh the components by a specified container, or open a new window/dialog with the resulting XML.

This action is applied to the button listener below.

```
<BUTTON NAME="helloButton" TEXT="Hello World Dialog">
 <LISTENER EVENT="ActionListener.actionPerformed">

   <CONTROLLER-ACTION COMPONENT="helloPanel" OPERATION="Open"
URL="http://localhost:8080/SwingML/servlet/SayHelloServlet"/>

 </LISTENER>

 </BUTTON>
```

When clicked the specified URL is invoked that dynamically generates XML on the server, the resulting XML is displayed in a new Dialog Window. The resulting UI and JSP with XML tags are shown below.




```
<!DOCTYPE PANEL SYSTEM "WebContent/SwingML.dtd">
<DIALOG NAME="sayHelloDlg" WIDTH="200" HEIGHT="100">
<PANEL NAME="helloPanel" LAF="Windows" LAYOUT="GridLayout" ROWS="1"
COLS="1">

<LABEL NAME="label" FONT-STYLE="BOLD" TEXT="Hello at...
<%=request.getAttribute("time")%>"  />


</PANEL>
</DIALOG>
```