



AltioLive JavaScript Support

A description of the proof-of-concept implementation of the Java SE 6 scripting features in AltioLive..

Document Version 1.0

Author : Jim Crossley

Introduction

Altio has provided the ability to call JavaScript functions via the ACTJAVASCRIPT action for some time, but these functions had to be in the HTML page that contained the Altio applet, and could only communicate with Altio via function parameters and calling the processDataXML method on Altio.

The release of Java 6 included the javax.script API (JSR 223) and a Rhino based JavaScript engine reference implementation. This allows JavaScript code to be executed within the same process and virtual machine as a Java application. The new features of Java 6 provide a simple way to extend Altio with JavaScript by embedding JavaScript code in the Altio view file itself.

Requirements

The source code mentioned in this document is included in the controlX library, available as part of the Altio 5.0.1 release. The source code itself can be used in all versions of Altio, apart from one line that is Altio 5.0.1 specific (indicated by a source code comment).

You will need to be using a Java SE 6 JVM as your browser plug-in to run this code, as the Java scripting features are only found in this release.

JavaScript in Altio

As the new features are only found in the latest release of Java, adding support in the main product is not ideal, as Altio is targeting a wide variety of JVM versions.

A better way is to write a custom control to invoke the new JavaScript functionality in response to a custom ACTION. This simple, non-visual control, supports an ACTEXECSCRIPT action that will execute whatever JavaScript is embedded as the text value of the ACTEXECSCRIPT block, e.g.

```
<ACTEXECSCRIPT CONTROL="script1">
<![CDATA[
    println("Hello from JavaScript");
]]>
</ACTEXECSCRIPT>
```

All that this example will do is write *"Hello from JavaScript"* to the Java console. This is interesting, but executing JavaScript from Altio is nothing new.

The benefit of using the new Java 6 features is that the custom control can pass a reference to Altio objects to the JavaScript environment, allowing JavaScript code to use and modify internal Altio objects. These objects are the same ones available to Altio custom control developers, and so using JavaScript like this provides an easier way to implement small blocks of functionality for Altio applications without having to develop a custom control.

Custom control code

The interesting part of the new JavaScript custom control code is shown below:

```
public int applyAction(ActionRule action) {
    String name = action.getName();
    if (name.equals("ACTEXECSCRIPT")) {
        ScriptEngineManager mgr = new ScriptEngineManager();

        ScriptEngine jsEngine = mgr.getEngineByName("JavaScript");
        final String toExec = action.getActionElement().getText();
        try {
            // Pass objects to JavaScript
            jsEngine.put("altioUI", getUIController());
            jsEngine.put("parent", action.getControl()); // 5.01 ONLY
            jsEngine.put("dataRoot", getDataManager().getRootNode());
            jsEngine.put("this", this);
            jsEngine.eval(toExec);
        }
        catch (ScriptException ex) {
            ex.printStackTrace();
        }
    }
    return ActionRule.ACTIONS_CONTINUE;
}
```

This method handles the ACTEXECSCRIPT block, with the lines in red doing the work of loading the JavaScript engine with Altio objects and then executing the text contained in the ACTEXECSCRIPT block.

Altio objects

In the above example, the Altio objects passed in are:

altioUI	An instance of com.altio.client.UIController. The UIController handles the Altio view file management of Windows, Controls etc.
parent	An instance of com.altio.client.control.Control that is the control that generated the action (e.g. a Button if the action is triggered by a button click)
dataRoot	An instance of com.altio.xml.XmlElement that is the root <DATA> element in the local data cache of your application.
this	An instance of com.altio.client.control.Control that is the Altio scripting custom control executing the JavaScript.

Documentation on the methods available for each of these objects is shipped with Altio.

JavaScript examples

1. Changing the caption of a control:

```
<ACTEXECSCRIPT CONTROL="script1">
<![CDATA[
    parent.setProperty("CAPTION", "From JavaScript", null, null);
]]>
</ACTEXECSCRIPT>
```

2. Counting data elements and navigating the XML data tree.

```
<ACTEXECSCRIPT CONTROL="script1">
<![CDATA[
    var data = dataRoot;
    println("We have a data root with " + data.numChildren(true) + "
children");
    var elem = dataRoot.getFirstChild();
    var count = 0;
    while (elem != null) {
        count++;
        elem = elem.getNextSibling();
    }
    println("Counted " + count + " data elements");
]]>
</ACTEXECSCRIPT>
```

3. Changing an application dynamically.

This example widens the window, adds a new multi-line text control and sets its contents to be the application configuration XML.

```
<ACTEXECSCRIPT CONTROL="script1">
<![CDATA[
    // Find an existing Label control by name
    var label = altioUI.getViewXml().findNode("LABEL", "NM",
"JAVASCRIPT_OUTPUT");
    // Clone the XML Element for that control
    var newLabel = label.cloneElement(true);

    // Change our cloned LABEL into a TEXT control
    newLabel.setName("TEXT");
    newLabel.setAttr("NM", "NewText");
    newLabel.setAttr("Y", "100");
    newLabel.setAttr("H", "300");
    newLabel.setAttr("W", "500");
    newLabel.setAttr("WRAP", "Y");
    newLabel.setAttr("CAPTION", "Config XML");

    // Make window wider
    var win = parent.getWindow();
    win.setProperty("W", "700", null);

    // Create a new Altio control from our cloned config
    var panel = parent.getPanel();
    var newLabelControl = altioUI.createControl(win, panel, newLabel, null);

    // Add our new control to the panel
    panel.addControl(newLabelControl);

    // Put the Altio config XML into our new text control
    var configString = altioUI.getViewXml().toCleanString();

    newLabelControl.setProperty("VALUE", configString, null, null);
]]>
</ACTEXECSCRIPT>
```

Conclusion

Once the custom control to invoke the JavaScript engine is complete, you can then reuse this control in your application to run whatever JavaScript code you need, which could be simpler than developing Altio custom controls to provide the same functionality, as there is no need to program in Java or to compile and deploy code as a custom control, it can just be embedded in an XML file.

The fact that your application will need to run in a Java 6 JVM may mean that you cannot rely on JavaScript functionality for your application if you have users on older versions of the JVM. However if you can guarantee that you users are running the latest version Java then it can be a simple and effective way to enhance the core functionality of Altio.

At the moment you may have to use an external editor to add the JavaScript code as the text of your ACTEXECSCRIPT block, as the Altio development tools do not allow you to do this easily (you can right click on an Altio window and choose "View Source" to edit the XML directly though), but we will consider adding better support for both direct XML editing and JavaScript in future releases of the Altio Designer.