VelocityValidationToolHack

A first pass to make the client-side javascript validation portion of the Struts Validator library work using a Velocity Tool. This should work with the last Struts Validator build for Struts 1.0 (http://home.earthlink.net/~dwinterfeldt/ -> downloads -> validator20010702.zip)

Seting up the Validator jar and validation.xml config is beyond the scope of this doc (basically, if you already know how to use the Validator libs in Struts 1.0, then this tool will allow you to use the client-side validation with Velocity templates.)

Template example:

```
  <!-- javascript form validation -->
  #set ($foo = $validator.setFormName("nameofyourform"))
  $validator.javascript <-- spits out the dynamic javascript
```

Toolbox configuration

```
<tool>
       <key>validator</key>
       <scope>request</scope>
       <class>package org.apache.velocity.tools.struts.ValidatorTool</class>
    </tool>
```

This has been tested in a limited fashion for Struts 1.0, but shortly after it is committed, there are plans to upgrade it to Struts 1.1 and use the commons-validator packages. (This is where Dave Winterfeldts Struts Validator was moved.)

Cheers, Timo

```
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 * ====================================================================
 *
 * This software consists of voluntary contributions made by many
 * individuals on behalf of the Apache Software Foundation.  For more
 * information on the Apache Software Foundation, please see
 * <http://www.apache.org/>.
 */
package org.apache.velocity.tools.struts;

import com.wintecinc.struts.validation.*;
import com.wintecinc.struts.action.ValidatorServlet;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.util.MessageResources;
import org.apache.velocity.tools.view.context.ViewContext;
import org.apache.velocity.tools.view.tools.ViewTool;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import javax.servlet.jsp.JspException;
import java.util.*;


/**
 * View tool to work with the Struts Validator to produce client side javascript validation.
 *
 * <p>This is a quick copy/paste of the JavascriptValidator Tag
 * from Dave Winterfeldt's Struts 1.0 validator library Struts-Validator20010702.jar</p>
 *
 * @version $Id: $
 */
public class ValidatorTool implements ViewTool {

    // ------------------------------------------- Properties ---------------

    /**
     * A reference to the ServletContext
     */
    protected ServletContext application;


    /**
     * A reference to the HttpServletRequest.
     */
    protected HttpServletRequest request;


    /**
     * A reference to the HttpSession.
     */
    protected HttpSession session;


    /**
     * A reference to the Struts message resources.
     */
    protected MessageResources resources;


    /**
     * A reference to the user's locale.
     */
    protected Locale locale;
```

```
/**
 * A reference to the queued action messages.
 */
protected ActionErrors errors;


/**
 * The servlet context attribute key for our resources.
 */
protected String bundle = Action.MESSAGES_KEY;


/**
 * The default locale on our server.
 */
protected static Locale defaultLocale = Locale.getDefault();

/**
 * The name of the form that corresponds with the action name
 * in struts-config.xml.  This won't be needed if this is incorporated
 * as an option in the html:form tag (ex: javascript="true").
 */
protected String formName = null;

/**
 * The page number we are on in a multi-part form.  This won't be needed
 * if this is incorporated as an option in the html:form tag (should be able to be set
 * or retrieved from scope).
 */
protected int page = 0;

/**
 * This will be used as is for the Javascript validation method name if it has a value.  This is
 * the method name of the main Javascript method that the form calls to perform validations.
 */
protected String methodName = null;

/**
 * The static JavaScript methods will only be printed if this is set to "true".
 */
protected String staticJavascript = "true";

/**
 * The dynamic JavaScript objects will only be generated if this is set to "true".
 */
protected String dynamicJavascript = "true";

/**
 * The src attribute for html script element (used to include an external script resource).
 */
protected String src = null;

// ------------------------------------------- Constructors -------------

/**
 * Default constructor. Tool must be initialized before use.
 */
public ValidatorTool() {
}


/**
 * Initializes this tool.
 *
 * @param obj the current ViewContext
 * @throws IllegalArgumentException if the param is not a ViewContext
 */
public void init(Object obj) {
    if (!(obj instanceof ViewContext)) {
```

```
            throw new IllegalArgumentException("Tool can only be initialized with a ViewContext");
        }

        ViewContext context = (ViewContext) obj;
        this.request = context.getRequest();
        this.session = request.getSession(false);
        this.application = context.getServletContext();

        resources = StrutsUtils.getMessageResources(application);
        locale = StrutsUtils.getLocale(request, session);
        if (null == locale) {
            locale = defaultLocale;
        }
        errors = StrutsUtils.getActionErrors(request);
    }


    // --------------------------------------------- View Helpers -------------

    public String getFormName() {
        return formName;
    }

    public void setFormName(String formName) {
        this.formName = formName;
    }

    public int getPage() {
        return page;
    }

    public void setPage(int page) {
        this.page = page;
    }

    public String getMethod() {
        return methodName;
    }

    public void setMethod(String methodName) {
        this.methodName = methodName;
    }

    public String getStaticJavascript() {
        return staticJavascript;
    }

    public void setStaticJavascript(String staticJavascript) {
        this.staticJavascript = staticJavascript;
    }

    public String getDynamicJavascript() {
        return dynamicJavascript;
    }

    public void setDynamicJavascript(String dynamicJavascript) {
        this.dynamicJavascript = dynamicJavascript;
    }

    public String getSrc() {
        return src;
    }

    public void setSrc(String src) {
        this.src = src;
    }

    /**
     * Render the JavaScript for to perform validations based on the form name.
     *
     * @exception javax.servlet.jsp.JspException if a JSP exception has occurred
```

```
 */
public String getJavascript() throws JspException {
    StringBuffer results = new StringBuffer();

    ValidatorResources validatorResources =
            (ValidatorResources) application.getAttribute(ValidatorServlet.VALIDATOR_KEY);

    Form form = null;
    form = validatorResources.get(locale, formName);
    if (form != null && "true".equals(dynamicJavascript)) {

        List lActions = new ArrayList();
        List lActionMethods = new ArrayList();

        // Get List of actions for this Form
        for (Iterator i = form.getFields().iterator(); i.hasNext();) {
            Field field = (Field) i.next();

            for (Iterator x = field.getDependencies().iterator(); x.hasNext();) {
                Object o = x.next();

                if (o != null && !lActionMethods.contains(o))
                    lActionMethods.add(o);
            }

        }

        // Create list of ValidatorActions based on lActionMethods
        for (Iterator i = lActionMethods.iterator(); i.hasNext();) {
            ValidatorAction va = validatorResources.getValidatorAction((String) i.next());

            String javascript = va.getJavascript();
            if (javascript != null && javascript.length() > 0)
                lActions.add(va);
            else
                i.remove();
        }

        Collections.sort(lActions, new Comparator() {
            public int compare(Object o1, Object o2) {
                ValidatorAction va1 = (ValidatorAction) o1;
                ValidatorAction va2 = (ValidatorAction) o2;

                if ((va1.getDepends() == null || va1.getDepends().length() == 0) &&
                        (va2.getDepends() == null || va2.getDepends().length() == 0)) {
                    return 0;
                } else if ((va1.getDepends() != null && va1.getDepends().length() > 0) &&
                        (va2.getDepends() == null || va2.getDepends().length() == 0)) {
                    return 1;
                } else if ((va1.getDepends() == null || va1.getDepends().length() == 0) &&
                        (va2.getDepends() != null && va2.getDepends().length() > 0)) {
                    return -1;
                } else {
                    return va1.getDependencies().size() - va2.getDependencies().size();
                }
            }
        });

        String methods = null;
        for (Iterator i = lActions.iterator(); i.hasNext();) {
            ValidatorAction va = (ValidatorAction) i.next();

            if (methods == null)
                methods = va.getMethod() + "(form)";
            else
                methods += " && " + va.getMethod() + "(form)";
        }

        String javascriptBegin = getJavascriptBegin(methods);
        results.append(javascriptBegin);
```

```
            for (Iterator i = lActions.iterator(); i.hasNext();) {
                ValidatorAction va = (ValidatorAction) i.next();
                String jscriptVar = null;

                results.append("         function " + va.getName() + " () { \n");
                for (Iterator x = form.getFields().iterator(); x.hasNext();) {
                    Field field = (Field) x.next();

                    if (field.getPage() == page && field.isDependency(va.getName())) {
                        String message = ValidatorUtil.getMessage(resources, locale, va, field);
                        message = (message != null ? message : "");

                        jscriptVar = getNextVar(jscriptVar);

                        results.append("            this." + jscriptVar + " = new Array(\"" + field.
getProperty() + "\", \"" + message + "\", ");


                        results.append("new Function (\"varName\", \"");

                        Map hVars = field.getVars();
                        // Loop through the field's variables.
                        for (Iterator iVars = hVars.entrySet().iterator(); iVars.hasNext();) {
                            Map.Entry entry = (Map.Entry) iVars.next();
                            String varKey = (String) entry.getKey();
                            Var var = (Var) entry.getValue();
                            String varValue = var.getValue();
                            String jsType = var.getJsType();

                            if (Var.JSTYPE_INT.equalsIgnoreCase(jsType))
                                results.append("this." + varKey + "=" + ValidatorUtil.replace(varValue, "\\",
"\\\\") + "; ");
                            else if (Var.JSTYPE_REGEXP.equalsIgnoreCase(jsType))
                                results.append("this." + varKey + "=/" + ValidatorUtil.replace(varValue, "\\",
"\\\\") + "/; ");
                            else if (Var.JSTYPE_STRING.equalsIgnoreCase(jsType))
                                results.append("this." + varKey + "='" + ValidatorUtil.replace(varValue, "\\",
"\\\\") + "'; ");
                            // So everyone using the latest format doesn't need to change their xml files
immediately.
                            else if ("mask".equalsIgnoreCase(varKey))
                                results.append("this." + varKey + "=/" + ValidatorUtil.replace(varValue, "\\",
"\\\\") + "/; ");
                            else
                                results.append("this." + varKey + "='" + ValidatorUtil.replace(varValue, "\\",
"\\\\") + "'; ");
                        }

                        results.append(" return this[varName];\"));\n");
                    }
                }
                results.append("         } \n\n");
            }


        if ("true".equals(staticJavascript))
            results.append(getJavascriptStaticMethods(validatorResources));

        if ("true".equals(dynamicJavascript))
            results.append(getJavascriptEnd());

        }

        return results.toString();

    }


    protected String getJavascriptBegin(String methods) {
        StringBuffer sb = new StringBuffer();
        String name = formName.substring(0, 1).toUpperCase() + formName.substring(1, formName.length());
```

```
    if (src != null) {
        sb.append("<SCRIPT LANGUAGE=\"Javascript1.1\" src=\"" + src + "\"> \n");
    } else {
        sb.append("<SCRIPT LANGUAGE=\"Javascript1.1\"> \n");
    }

    sb.append("<!-- Begin \n");
    sb.append("\n         var bCancel = false; \n\n");

    if (methodName == null || methodName.length() == 0) {
        sb.append("         function validate" + name + "(form) { \n");
    } else {
        sb.append("         function " + methodName + "(form) { \n");
    }
    sb.append("              if (bCancel) \n");
    sb.append("                 return true; \n");
    sb.append("              else \n");
    sb.append("                 return " + methods + "; \n");
    sb.append("         } \n\n");

    return sb.toString();
}

protected String getJavascriptStaticMethods(ValidatorResources resources) {
    StringBuffer sb = new StringBuffer();

    sb.append("\n\n");

    for (Iterator i = resources.getValidatorActions().values().iterator(); i.hasNext();) {
        ValidatorAction va = (ValidatorAction) i.next();
        if (va != null) {
            String javascript = va.getJavascript();
            if (javascript != null && javascript.length() > 0)
                sb.append(javascript + "\n");
        }
    }

    return sb.toString();
}

protected String getJavascriptEnd() {
    StringBuffer sb = new StringBuffer();

    sb.append("\n");
    sb.append("//  End -->\n");
    sb.append("</SCRIPT>\n\n");

    return sb.toString();
}

/**
 * The value <code>null</code> will be returned at the end of the sequence.
 *     ex: "zz" will return <code>null</code>
 */
private String getNextVar(String input) {
    if (input == null)
        return "aa";

    input = input.toLowerCase();

    for (int i = input.length(); i > 0; i--) {
        int pos = i - 1;

        char c = input.charAt(pos);
        c++;

        if (c <= 'z') {
            if (i == 0)
                return c + input.substring(pos, input.length());
            else if (i == input.length())
```

```
                    return input.substring(0, pos) + c;
                else
                    return input.substring(0, pos) + c + input.substring(pos, input.length() - 1);
            } else {
                input = replaceChar(input, pos, 'a');
            }

        }

        return null;

    }

    /**
     * Replaces a single character in a <code>String</code>
     */
    private String replaceChar(String input, int pos, char c) {
        if (pos == 0)
            return c + input.substring(pos, input.length());
        else if (pos == input.length())
            return input.substring(0, pos) + c;
        else
            return input.substring(0, pos) + c + input.substring(pos, input.length() - 1);
    }


}
```