Inspection

Version 1.0

Giorgio Pea(Mat. 853872), Andrea Sessa(Mat. 850082)

5/1/2016

Contents

1 Introduction

2 Classes

Included in this section the two java classes subjected to the analisys.

File: /appserver/web/web-core/src/main/java/org/apache/catalina/ssi/**SSIServlet.java**

Methods under inspection:

- ***init()***
- ***requestHandler(HttpServletRequest req, HttpServletResponse res)***
- ***processSSI(HttpServletRequest req , HttpServletResponse res , URL resource)***

```
package org.apache.catalina.ssi;


import org.apache.catalina.Globals;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.net.URL;
import java.net.URLConnection;
import java.util.Locale;
/**
* Servlet to process SSI requests within a webpage. Mapped to a path from
* within web.xml.
*
* @author Bip Thelin
* @author Amy Roh
* @author Dan Sandberg
* @author David Becker
* @version $Revision: 1.4 $, $Date: 2007/05/05 05:32:20 $
*/
public class SSIServlet extends HttpServlet {
    /** Debug level for this servlet. */
    protected int debug = 0;
    /** Should the output be buffered. */
    protected boolean buffered = false;
```

```
/** Expiration time in seconds for the doc. */
protected Long expires = null;
/** virtual path can be webapp-relative */
protected boolean isVirtualWebappRelative = false;
/** Input encoding. If not specified, uses platform default */
protected String inputEncoding = null;
/** Output encoding. If not specified, uses platform default */
protected String outputEncoding = "UTF-8";


//----------------- Public methods.
/**
* Initialize this servlet.
*
* @exception ServletException
*                if an error occurs
*/
public void init() throws ServletException {

    if (getServletConfig().getInitParameter("debug") != null)
        debug = Integer.parseInt(getServletConfig().getInitParameter(
            "debug"));

    isVirtualWebappRelative =
        Boolean.parseBoolean(getServletConfig().getInitParameter("
            isVirtualWebappRelative"));

    if (getServletConfig().getInitParameter("expires") != null)
        expires = Long.valueOf(getServletConfig().getInitParameter("
            expires"));

    buffered = Boolean.parseBoolean(getServletConfig().
        getInitParameter("buffered"));

    inputEncoding = getServletConfig().getInitParameter("
        inputEncoding");

    if (getServletConfig().getInitParameter("outputEncoding") != null
        )
        outputEncoding = getServletConfig().getInitParameter("
            outputEncoding");

    if (debug > 0)
        log("SSIServlet.init() SSI invoker started with 'debug'=" +
            debug);

}


/**
* Process and forward the GET request to our <code>requestHandler()</
    code>*
*
```

```
* @param req
*            a value of type 'HttpServletRequest'
* @param res
*            a value of type 'HttpServletResponse'
* @exception IOException
*                if an error occurs
* @exception ServletException
*                if an error occurs
*/
public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws IOException, ServletException {
    if (debug > 0) log("SSIServlet.doGet()");
    requestHandler(req, res);
}


/**
* Process and forward the POST request to our
* <code>requestHandler()</code>.
*
* @param req
*            a value of type 'HttpServletRequest'
* @param res
*            a value of type 'HttpServletResponse'
* @exception IOException
*                if an error occurs
* @exception ServletException
*                if an error occurs
*/
public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws IOException, ServletException {
    if (debug > 0) log("SSIServlet.doPost()");
    requestHandler(req, res);
}


/**
* Process our request and locate right SSI command.
*
* @param req
*            a value of type 'HttpServletRequest'
* @param res
*            a value of type 'HttpServletResponse'
*/
protected void requestHandler(HttpServletRequest req,
        HttpServletResponse res) throws IOException, ServletException
            {
    ServletContext servletContext = getServletContext();
    String path = SSIServletRequestUtil.getRelativePath(req);
    if (debug > 0)
        log("SSIServlet.requestHandler()\n" + "Serving␣"
                + (buffered?"buffered␣":"unbuffered␣") + "resource␣'"
                + path + "'");
```

```
        // Exclude any resource in the /WEB-INF and /META-INF
            subdirectories
        // (the "toUpperCase()" avoids problems on Windows systems)
        if (path == null || path.toUpperCase(Locale.ENGLISH).startsWith("
            /WEB-INF")
                || path.toUpperCase(Locale.ENGLISH).startsWith("/META-INF
                    ")) {
            res.sendError(HttpServletResponse.SC_NOT_FOUND, path);
            log("Can't serve file: " + path);
            return;
        }
        URL resource = servletContext.getResource(path);
        if (resource == null) {
            res.sendError(HttpServletResponse.SC_NOT_FOUND, path);
            log("Can't find file: " + path);
            return;
        }
        String resourceMimeType = servletContext.getMimeType(path);
        if (resourceMimeType == null) {
            resourceMimeType = "text/html";
        }
        res.setContentType(resourceMimeType + ";charset=" +
            outputEncoding);
        if (expires != null) {
            res.setDateHeader("Expires", (new java.util.Date()).getTime()
                    + expires.longValue() * 1000);
        }
        req.setAttribute(Globals.SSI_FLAG_ATTR, "true");
        processSSI(req, res, resource);
    }


    protected void processSSI(HttpServletRequest req, HttpServletResponse
         res,
            URL resource) throws IOException {
        SSIExternalResolver ssiExternalResolver =
            new SSIServletExternalResolver(getServletContext(), req, res,
                    isVirtualWebappRelative, debug, inputEncoding);
        SSIProcessor ssiProcessor = new SSIProcessor(ssiExternalResolver,
                debug);
        PrintWriter printWriter = null;
        StringWriter stringWriter = null;
        if (buffered) {
            stringWriter = new StringWriter();
            printWriter = new PrintWriter(stringWriter);
        } else {
            printWriter = res.getWriter();
        }

        URLConnection resourceInfo = resource.openConnection();
        InputStream resourceInputStream = resourceInfo.getInputStream();
        String encoding = resourceInfo.getContentEncoding();
        if (encoding == null) {
```

```
                encoding = inputEncoding;
            }
            InputStreamReader isr;
            if (encoding == null) {
                isr = new InputStreamReader(resourceInputStream);
            } else {
                isr = new InputStreamReader(resourceInputStream, encoding);
            }
            BufferedReader bufferedReader = new BufferedReader(isr);

            long lastModified = ssiProcessor.process(bufferedReader,
                    resourceInfo.getLastModified(), printWriter);
            if (lastModified > 0) {
                res.setDateHeader("last-modified", lastModified);
            }
            if (buffered) {
                printWriter.flush();
                String text = stringWriter.toString();
                res.getWriter().write(text);
            }
        }
    }
```

File: /appserver/web/web-core/src/main/java/org/apache/catalina/ssi/**SSIMediator.java**

Methods under inspection:

- ***substituteVariables(String val)***

```
package org.apache.catalina.ssi;


import org.apache.catalina.util.Strftime;
import org.apache.catalina.util.URLEncoder;

import java.io.IOException;
import java.util.Collection;
import java.util.Date;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Locale;
import java.util.Set;
import java.util.TimeZone;
import org.glassfish.grizzly.http.util.HttpUtils;

/**
* Allows the different SSICommand implementations to share data/talk to
    each
* other
*
* @author Bip Thelin
* @author Amy Roh
* @author Paul Speed
* @author Dan Sandberg
* @author David Becker
* @version $Revision: 1.5 $, $Date: 2007/05/05 05:32:20 $
*/
public class SSIMediator {
    protected final static String DEFAULT_CONFIG_ERR_MSG = "[an error
        occurred while processing this directive]";
    protected final static String DEFAULT_CONFIG_TIME_FMT = "%A, %d-%b-%Y
        %T %Z";
    protected final static String DEFAULT_CONFIG_SIZE_FMT = "abbrev";
    protected final static URLEncoder urlEncoder;
    protected String configErrMsg = DEFAULT_CONFIG_ERR_MSG;
    protected String configTimeFmt = DEFAULT_CONFIG_TIME_FMT;
    protected String configSizeFmt = DEFAULT_CONFIG_SIZE_FMT;
    protected String className = getClass().getName();
    protected SSIExternalResolver ssiExternalResolver;
    protected long lastModifiedDate;
    protected Strftime strftime;
    protected SSIConditionalState conditionalState = new
        SSIConditionalState();

    static {
        //We try to encode only the same characters that apache does
```

```
        urlEncoder = new URLEncoder();
        urlEncoder.addSafeCharacter(',');
        urlEncoder.addSafeCharacter(':');
        urlEncoder.addSafeCharacter('-');
        urlEncoder.addSafeCharacter('_');
        urlEncoder.addSafeCharacter('.');
        urlEncoder.addSafeCharacter('*');
        urlEncoder.addSafeCharacter('/');
        urlEncoder.addSafeCharacter('!');
        urlEncoder.addSafeCharacter('~');
        urlEncoder.addSafeCharacter('\'');
        urlEncoder.addSafeCharacter('(');
        urlEncoder.addSafeCharacter(')');
    }


    public SSIMediator(SSIExternalResolver ssiExternalResolver, long
        lastModifiedDate, int debug) { ... }

    public void setConfigErrMsg(String configErrMsg) { ... }

    public void setConfigTimeFmt(String configTimeFmt) { ... }

    public void setConfigTimeFmt(String configTimeFmt, boolean
        fromConstructor) { ... }

    public void setConfigSizeFmt(String configSizeFmt) { ... }

    public String getConfigErrMsg() { ... }

    public String getConfigTimeFmt() { ... }

    public String getConfigSizeFmt() { ... }

    public SSIConditionalState getConditionalState() { ... }


    public Collection<String> getVariableNames() {
        Set<String> variableNames = new HashSet<String>();
        //These built-in variables are supplied by the mediator ( if not
        // over-written by
        // the user ) and always exist
        variableNames.add("DATE_GMT");
        variableNames.add("DATE_LOCAL");
        variableNames.add("LAST_MODIFIED");
        ssiExternalResolver.addVariableNames(variableNames);
        //Remove any variables that are reserved by this class
        Iterator<String> iter = variableNames.iterator();
        while (iter.hasNext()) {
            String name = iter.next();
            if (isNameReserved(name)) {
                iter.remove();
            }
```

```
        }
        return variableNames;
    }


    public long getFileSize(String path, boolean virtual) throws
        IOException {
        return ssiExternalResolver.getFileSize(path, virtual);
    }


    public long getFileLastModified(String path, boolean virtual)
            throws IOException {
        return ssiExternalResolver.getFileLastModified(path, virtual);
    }


    public String getFileText(String path, boolean virtual) throws
        IOException {
        return ssiExternalResolver.getFileText(path, virtual);
    }


    protected boolean isNameReserved(String name) {
        return name.startsWith(className + ".");
    }


    public String getVariableValue(String variableName) {
        return getVariableValue(variableName, "none");
    }


    public void setVariableValue(String variableName, String
        variableValue) {
        if (!isNameReserved(variableName)) {
            ssiExternalResolver.setVariableValue(variableName,
                variableValue);
        }
    }


    public String getVariableValue(String variableName, String encoding)
        {
        String lowerCaseVariableName = variableName.toLowerCase(Locale.
            ENGLISH);
        String variableValue = null;
        if (!isNameReserved(lowerCaseVariableName)) {
            //Try getting it externally first, if it fails, try getting
                the
            // 'built-in'
            // value
            variableValue = ssiExternalResolver.getVariableValue(
```

```
                variableName);
            if (variableValue == null) {
                variableName = variableName.toUpperCase(Locale.ENGLISH);
                variableValue = ssiExternalResolver
                        .getVariableValue(className + "." + variableName)
                            ;
            }
            if (variableValue != null) {
                variableValue = encode(variableValue, encoding);
            }
        }
        return variableValue;
    }


    /**
    * Applies variable substitution to the specified String and returns
        the
    * new resolved string.
    */
    public String substituteVariables(String val) {
        // If it has no references or HTML entities then no work
        // need to be done
        if (val.indexOf('$') < 0 && val.indexOf('&') < 0) return val;

        // HTML decoding
        val = val.replace("&lt;", "<");
        val = val.replace("&gt;", ">");
        val = val.replace("&quot;", "\"");
        val = val.replace("&amp;", "&");

        StringBuilder sb = new StringBuilder(val);
        int charStart = sb.indexOf("&#");
        while (charStart > -1) {
            int charEnd = sb.indexOf(";", charStart);
            if (charEnd > -1) {
                char c = (char) Integer.parseInt(
                        sb.substring(charStart + 2, charEnd));
                sb.delete(charStart, charEnd + 1);
                sb.insert(charStart, c);
                charStart = sb.indexOf("&#");
            } else {
                break;
            }
        }

        for (int i = 0; i < sb.length();) {
            // Find the next $
            for (; i < sb.length(); i++) {
                if (sb.charAt(i) == '$') {
                    i++;
                    break;
                }
```

```
            }
            if (i == sb.length()) break;
            // Check to see if the $ is escaped
            if (i > 1 && sb.charAt(i - 2) == '\\') {
                sb.deleteCharAt(i - 2);
                i--;
                continue;
            }
            int nameStart = i;
            int start = i - 1;
            int end = -1;
            int nameEnd = -1;
            char endChar = ' ';
            // Check for {} wrapped var
            if (sb.charAt(i) == '{') {
                nameStart++;
                endChar = '}';
            }
            // Find the end of the var reference
            for (; i < sb.length(); i++) {
                if (sb.charAt(i) == endChar) break;
            }
            end = i;
            nameEnd = end;
            if (endChar == '}') end++;
            // We should now have enough to extract the var name
            String varName = sb.substring(nameStart, nameEnd);
            String value = getVariableValue(varName);
            if (value == null) value = "";
            // Replace the var name with its value
            sb.replace(start, end, value);
            // Start searching for the next $ after the value
            // that was just substituted.
            i = start + value.length();
        }
        return sb.toString();
    }


    protected String formatDate(Date date, TimeZone timeZone) { ... }


    protected String encode(String value, String encoding) { ... }


    public void log(String message) {
        ssiExternalResolver.log(message, null);
    }


    public void log(String message, Throwable throwable) {
        ssiExternalResolver.log(message, throwable);
    }
```

```
    protected void setDateVariables(boolean fromConstructor) { ... }
}
```

3 Functional Role

4 Issues

5 Additional Considerations