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)***

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

Methods under inspection:

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

3 Functional Role

In this section are included some information about the functioning of tha analyzed classes and methods.

3.1 SSIServlet.java

From the Javadoc:

```
78 /**
79  * Servlet to process SSI requests within a webpage. Mapped to a path from
80  * within web.xml.
81  *
```

This class represents a Java EE servlet used to process requests that include some SSI instruction.
SSI(Server Side Include) that is a simple interpreted server-side scripting language. The most frequent use of SSI is to include the contents of one or more files into a web page on a web server.

- *Init()*
- *requestHandler()*
  From the inspection of the code this function is only called when the servelet receives a http Get or Post request. The javadoc for the method, included in the code, states:

```
173     /**
174      * Process our request and locate right SSI command.
175      *
176      * @param req
177      *            a value of type 'HttpServletRequest'
178      * @param res
179      *            a value of type 'HttpServletResponse'
180      */
```

Hence the method accepts as parameters a HttpServletRequest,the incoming request, and a HttpServletResponse that is a reference to the response.

Now the objective of the method is to retrieve the correct resource from the servelet context. If the debug level is greater than zero then log a message into the logger for debug purposes.

```
183         ServletContext servletContext = getServletContext();
184         String path = SSIServletRequestUtil.getRelativePath(req);
185         if (debug > 0)
186             log("SSIServlet.requestHandler()\n" + "Serving "
187                     + (buffered?"buffered ":"unbuffered ") + "resource '"
188                     + path + "'");
```

The comment is very clear: it checks if the resource is either in the ‘WEB-INF‘ or ‘META-INF‘ subdirectories; if so the function return with an error code.

```
189         // Exclude any resource in the /WEB-INF and /META-INF subdirectories
190         // (the "toUpperCase()" avoids problems on Windows systems)
191         if (path == null || path.toUpperCase(Locale.ENGLISH).startsWith("/WEB-INF")
192                 || path.toUpperCase(Locale.ENGLISH).startsWith("/META-INF")) {
193             res.sendError(HttpServletResponse.SC_NOT_FOUND, path);
194             log("Can't serve file: " + path);
195             return;
196         }
```

Here the fucntion tries to retrieve the URL to the resource; it also performs an existence check on the resource, if the resource doesn't exist the function return an error.

```
197         URL resource = servletContext.getResource(path);
198         if (resource == null) {
199             res.sendError(HttpServletResponse.SC_NOT_FOUND, path);
200             log("Can't find file: " + path);
201             return;
202         }
```

In the final part, the function starts to initialize the header of the HttpServletResponse by setting: the mime type, the encoding of the output text and the expiration time for the response(in seconds, see init()).
Finally the processSSI() function is invoked passing as parameters the original request, the reference to the response and the resource.

```
203         String resourceMimeType = servletContext.getMimeType(path);
204         if (resourceMimeType == null) {
205             resourceMimeType = "text/html";
206         }
207         res.setContentType(resourceMimeType + ";charset=" + outputEncoding);
208         if (expires != null) {
209             res.setDateHeader("Expires", (new java.util.Date()).getTime()
210                     + expires.longValue() * 1000);
211         }
212         req.setAttribute(Globals.SSI_FLAG_ATTR, "true");
213         processSSI(req, res, resource);
```

- *processSSI()*

3.2 SSIMediator.java

4 Issues

In this section is included a list of problems found during the ispection of the assigned code.

4.1 SSIServlet.java

General Considerations
In general the class lacks of documentation: comments and javadoc are not complete and where inserted are sometimes meaningless and very short.

- *init()*
- *requestHandler()*
  1. Checklist[8,9]: All indentations in the class are made by means of tabs
  2. Checklist[18]: The function is not fully commented, some instructions(lines 197 to 213) are not commented at all
  3. Checklist[11]: The conditional block

```
185         if (debug > 0)
186             log("SSIServlet.requestHandler()\n" + "Serving "
187                     + (buffered?"buffered ":"unbuffered ") + "resource '"
188                     + path + "'");
```

     uses no enclosing braces
  4. Checklist[33]: The declarations of variables in lines

```
197         URL resource = servletContext.getResource(path);

203         String resourceMimeType = servletContext.getMimeType(path);
```

  5. Checklist[40]: The lines

```
191         if (path == null || path.toUpperCase(Locale.ENGLISH).startsWith("/WEB-INF

198         if (resource == null) {

204         if (resourceMimeType == null) {

208         if (expires != null) {
```

     uses for comparation '==' instead of 'equals()'

- *processSSI()*

4.2 SSIMediator.java

- *substituteVariables()*

5 Additional Considerations

6 Appendix

6.1 Java Checklist

6.2 Statistics