Politecnico di Milano

Code inspection

"myTaxiService"

Nicolas Tagliabue(matr. 853097), Matteo Pagliari(matr. 854353)

January 5, 2016

Versione 1.0

Contents

1 Classes assigned

These are the classes the we were assigned to analyze, which are part of GlassFish 4.1.1

https://github.com/sernek/Project-SE2.git

Professor: Mirandola
Students: Nicolas Tagliabue, Matteo Pagliari

Methods:

- **Name:**createModuleInfo(Entry entry , CallbackHandler handler , String type , Map properties)
  Location:appserver/security/core-ee/src/main/java/com/sun/enterprise/security/jmac/config/GFServerConfigProvider.java
- **Name:**getEntry(String intercept , String id , MessagePolicy requestPolicy , MessagePolicy responsePolicy , String type)
  Location:appserver/security/core-ee/src/main/java/com/sun/enterprise/security/jmac/config/GFServerConfigProvider.java

2 Functional role

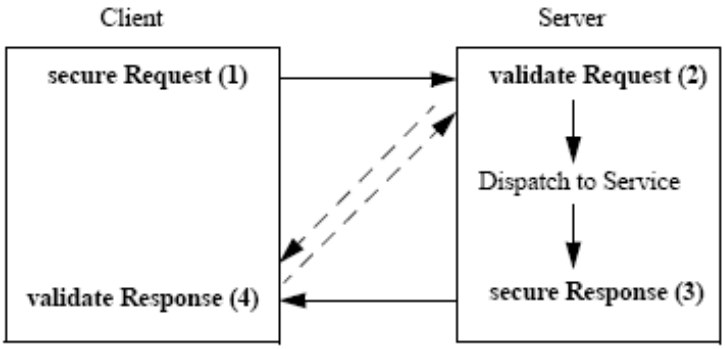
Our class implements the AuthConfigProvider interface whose job is to implement an authentication mechanism to allow messages exchange between a ServerAuthModule and a ClientAuthModule according to different protocols. Callers do not directly operate on authentication modules. Instead, they rely on a ClientAuthContext or ServerAuthContext to manage the invocation of modules. Because it implements this interface, our class is obliged to provide a public constructor with this signature:
public AuthConfigProviderImpl(Map properties, AuthConfigFactory factory)
and these 3 methods:

getClientAuthConfig(java.lang.String layer, java.lang.String appContext, javax.security.auth.callback.CallbackHandler handler)

getServerAuthConfig(java.lang.String layer, java.lang.String appContext, javax.security.auth.callback.CallbackHandler handler)

refresh()

To be more precise the GFserverConfigProvider class is instantiated by PipeHelper class in order to establish a secure connection between client and server in the context of JSR 196 (Java Authentication Service Provider Interface for Containers). JSR 196 defines a standard service-provider interface (SPI) for integrating authentication mechanism implementations in message processing runtimes. Here is a simple schema of the message processing model of JSR 196:



GFserverConfigProvider gets also instantiated by HttpServletHelper which is, quoting the Javadoc: "This is the realm adapter used to authenticate users and authorize access to web resources. The authenticate method is called by Tomcat to authenticate users ". Tomcat is a java application servlet container which is responsible for servlet creation, execution and destruction.

3 Issues found applying the checklist

3.1 Indention

9. No tabs are used to indent

```
243 ····· com.sun.enterprise.security.jauth.ServerAuthModule sam0 =
244 ····· (com.sun.enterprise.security.jauth.ServerAuthModule)
245 » » » newModule;
```

```
263 ····· } else if (newModule instanceof
264 » » ··· com.sun.enterprise.security.jauth.ClientAuthModule) {
```

```
266 ····· com.sun.enterprise.security.jauth.ClientAuthModule cam0 =
267 ····· (com.sun.enterprise.security.jauth.ClientAuthModule)
268 » » » newModule;
```

```
270 ····· AuthPolicy requestPolicy =
271 » » » new AuthPolicy(entry.getRequestPolicy());
272
273 » » ··· AuthPolicy responsePolicy =
274 » » » new AuthPolicy(entry.getResponsePolicy());
```

3.2 Wrapping Lines

15. Line break occurs after a comma or an operator.

```
343           if (logger.isLoggable(Level.FINE)) {
344               logger.fine("module config has no IDs configured for [" +
345                           intercept +
346                           "]");
347           }
```

```
362           if (logger.isLoggable(Level.FINE)) {
363               logger.fine("DD did not specify ID, " +
364                           "or DD-specified ID for [" +
365                           intercept +
366                           "] not found in config -- " +
367                           "attempting to look for default ID");
368           }
```

```
382               if (logger.isLoggable(Level.FINE)) {
383                   logger.fine("no default config ID for [" +
384                               intercept +
385                               "]");
386               }
```

```
396           if (logger.isLoggable(Level.FINE)) {
397               logger.fine("request type [" +
398                           type +
399                           "] does not match config type [" +
400                           idEntry.type +
401                           "]");
402           }
```

```
407       MessagePolicy reqP =
408           (requestPolicy != null || responsePolicy != null) ?
409           requestPolicy :
410           idEntry.requestPolicy;  //default;
```

```
412       MessagePolicy respP =
413           (requestPolicy != null || responsePolicy != null) ?
414           responsePolicy :
415           idEntry.responsePolicy;  //default;
```

```
430       if (logger.isLoggable(Level.FINE)) {
431           logger.fine("getEntry for: " + intercept + " -- " + id +
432                       "\n    module class: " + entry.moduleClassName +
433                       "\n    options: " + entry.options +
434                       "\n    request policy: " + entry.requestPolicy +
435                       "\n    response policy: " + entry.responsePolicy);
436       }
```

```
247                     AuthPolicy requestPolicy =
248                             (entry.getRequestPolicy() != null) ?
249                             new AuthPolicy(entry.getRequestPolicy()) : null;
250
251                     AuthPolicy responsePolicy =
252                             (entry.getResponsePolicy() != null) ?
253                             new AuthPolicy(entry.getResponsePolicy()) : null;
```

Other errors:

```
238                     sam.initialize(entry.getRequestPolicy(),
239                             entry.getResponsePolicy(), handler, map);
```

```
255                     sam0.initialize(requestPolicy, responsePolicy,
256                             handler, map);
```

```
276                     cam0.initialize(requestPolicy,responsePolicy,
277                             handler, map);
```

When you break a method call, you should align the new line with the first parameter

3.3 Java Source File

23. Check that the Javadoc is complete (i.e., it covers all classes and files part of the set of classes assigned to you).

```
292     /**
293      * Create an object of a given class.
294      * @param className
295      *
296      */
297     private static Object createObject(final String className) {
```

param className is missing description.

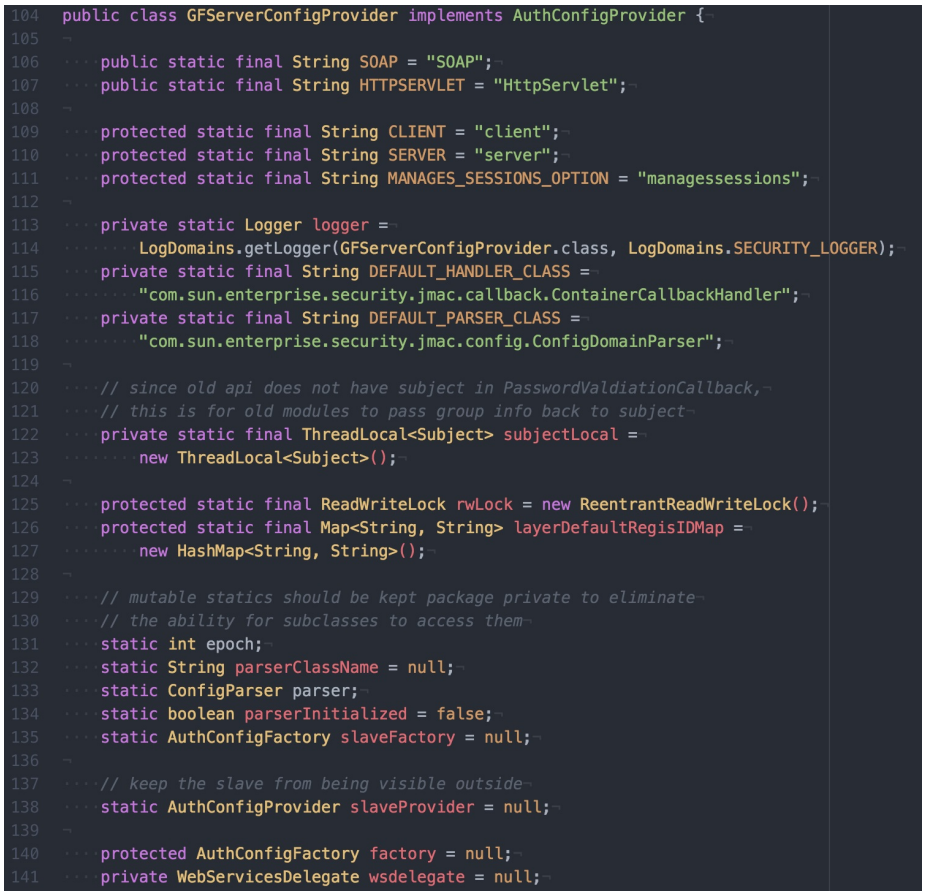
There is one public class (InterceptEntry) which does not have Javadoc.
Our assigned methods does not have an exhaustive Javadoc, even if it is not compulsory, because they are friendly.

3.4 Class and Interface Declarations

25. The class or interface declarations shall be in the following order:

1. class/interface documentation comment
2. class or interface statement
3. class/interface implementation comment, if necessary

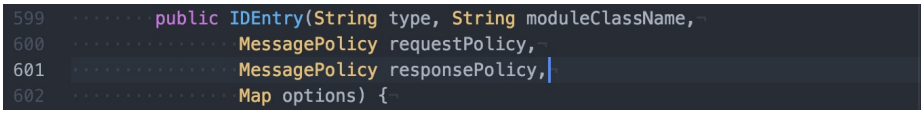
4. class (static) variables
    (a) first public class variables
    (b) next protected class variables
    (c) next package level (no access modifier)
    (d) last private class variables
5. instance variables
    (a) first public instance variables
    (b) next protected instance variables
    (c) next package level (no access modifier)
    (d) last private instance variables
6. constructors
7. methods



```
104 public class GFServerConfigProvider implements AuthConfigProvider {
105
106     public static final String SOAP = "SOAP";
107     public static final String HTTPSERVLET = "HttpServlet";
108
109     protected static final String CLIENT = "client";
110     protected static final String SERVER = "server";
111     protected static final String MANAGES_SESSIONS_OPTION = "managessessions";
112
113     private static Logger logger =
114         LogDomains.getLogger(GFServerConfigProvider.class, LogDomains.SECURITY_LOGGER);
115     private static final String DEFAULT_HANDLER_CLASS =
116         "com.sun.enterprise.security.jmac.callback.ContainerCallbackHandler";
117     private static final String DEFAULT_PARSER_CLASS =
118         "com.sun.enterprise.security.jmac.config.ConfigDomainParser";
119
120     // since old api does not have subject in PasswordValdiationCallback,
121     // this is for old modules to pass group info back to subject
122     private static final ThreadLocal<Subject> subjectLocal =
123         new ThreadLocal<Subject>();
124
125     protected static final ReadWriteLock rwLock = new ReentrantReadWriteLock();
126     protected static final Map<String, String> layerDefaultRegisIDMap =
127         new HashMap<String, String>();
128
129     // mutable statics should be kept package private to eliminate
130     // the ability for subclasses to access them
131     static int epoch;
132     static String parserClassName = null;
133     static ConfigParser parser;
134     static boolean parserInitialized = false;
135     static AuthConfigFactory slaveFactory = null;
136
137     // keep the slave from being visible outside
138     static AuthConfigProvider slaveProvider = null;
139
140     protected AuthConfigFactory factory = null;
141     private WebServicesDelegate wsdelegate = null;
```

Static attributes of line 125 and 126 should be placed near protected static attributes.

Static friendly attributes of line 131-138 should be placed between protected and private attributes.



```
599         public IDEntry(String type, String moduleClassName,
600                        MessagePolicy requestPolicy,
601                        MessagePolicy responsePolicy,
602                        Map options) {
```

Constructor should be placed before methods

27. Check that the code is free of duplicates, long methods, big classes, breaking encapsulation, as well as if coupling and cohesion are adequate.

```
1224        public void cleanSubject(MessageInfo messageInfo, Subject subject)
1225                throws AuthException {
1226            if (module != null) {
1227                module.cleanSubject(messageInfo, subject);
1228            } else if (oldModule != null) {
1229                oldModule.disposeSubject(subject, messageInfo.getMap());
1230            } else {
1231                throw new AuthException();
1232            }
1233        }
```

```
1292        public void cleanSubject(MessageInfo messageInfo, Subject subject)
1293                throws AuthException {
1294            if (module != null) {
1295                module.cleanSubject(messageInfo, subject);
1296            } else if (oldModule != null) {
1297                oldModule.disposeSubject(subject, messageInfo.getMap());
1298            } else {
1299    »   »   throw new AuthException();
1300            }
1301        }
```

3.5 Output Format

41. Check that displayed output is free of spelling and grammatical errors.

line 88 : backward and compatibility are misspelled
line 122 : validation is misspelled
line 734 : occurred is misspelled
line 749 : safety is misspelled
line 929 : authentication is misspelled
line 933 : occurred is misspelled
line 1019 : explicitly is misspelled

4 Any Other Problem

line 372-374 : it is better to use a getter method for getting a parameter.