Internal Project
-
Code Inspection

Davide Cremona (matr. 852365), Simone Deola (matr. 788181)

December 29, 2015

Contents

Chapter 1

Classes Assigned

Chapter 2

Functional Role of the Classes Assigned

Chapter 3

Issues Found

3.1 Class Issues

TODO - inserire commentino inizio sezione

- **Checklist C.01**

```java
90      private static final ResourceBundle rb =
            log.getResourceBundle();
```

"rb" does not mean anything.

```java
151     private LifecycleSupport lifecycle = new LifecycleSupport(this);
```

"lifecycle" does not suggest that is a lifecycleSupport.

```java
168     protected int debug = 0;
```

"debug" does not suggest that is a debug level variable.

```java
200     private NotificationBroadcasterSupport broadcaster = null;
```

"broadcaster" does not suggest that is a NOTIFICATION broadcaster.

```java
279     public int getDebug() {
```

Does not suggest that is the debug level.

```java
291     public void setDebug(int debug) {
```

Does not suggest that is the debug level.

```
745     protected ObjectName oname;
```

"oname" is meaningless.

- **Checklist C.07**

```
90       private static final Logger log = StandardServer.log;
91       private static final ResourceBundle rb =
             log.getResourceBundle();
```

```
138     private static final String info =
139         "org.apache.catalina.core.StandardService/1.0";
```

```
182     protected final Object connectorsMonitor = new Object();
```

Final attributes but not uppercase and separated by an underscore.

- **Checklist C.08**

  Lines 87 to 132, 203 to 214 and 355 to 374 have an additional space character at the beginning of the line. These spaces make the indentation not correct.

- **Checklist C.12**

```
77  /**
78   *  implementation of the <code>Service</code> interface. The
79   * associated Container is generally an instance of Engine, but
         this is
80   * not required.
81   *
82   * @author Craig R. McClanahan
83   */
84
85  public class StandardService
86          implements Lifecycle, Service
87   {
```

The line 84 is a blank line that should not divide the javadoc from the prototype of the method.

```
205      /**
206       * Construct a default instance of this class.
207       */
208
209      public StandardService() {
```

The line 208 is a blank line that should not divide the javadoc from the prototype of the method.

```
364     /**
365      * Set the <code>NotificationBroadcasterSupport</code> that
             sends notification for this Service
366      *
367      * @param broadcaster The new NotificationBroadcasterSupport
368      */
369
370     public void setBroadcaster(NotificationBroadcasterSupport
            broadcaster) {
```

The line 369 is a blank line that should not divide the javadoc from the prototype of the method.

The last line of the class (line 756) is useless.

- **Checklist C.15**

```
85  public class StandardService
86          implements Lifecycle, Service
```

Line break occurs after a space.

- **Checklist C.17**

All the class lines have an offset of one space character from their level of indentation.

- **Checklist C.23**

```
96      public static final String SERVICE_STARTED =
            "AS-WEB-CORE-00251";
```

```
102     public static final String STARTING_SERVICE =
            "AS-WEB-CORE-00252";
```

```
108     public static final String STOPPING_SERVICE =
            "AS-WEB-CORE-00253";
```

```
114     public static final String SERVICE_HAS_BEEN_INIT =
            "AS-WEB-CORE-00254";
```

```
122     public static final String ERROR_REGISTER_SERVICE_EXCEPTION =
            "AS-WEB-CORE-00255";
```

```java
    public static final String FAILED_SERVICE_INIT_EXCEPTION =
        "AS-WEB-CORE-00256";
```

No javadoc for public static attribute.

```java
  public ObjectName getContainerName() {
```

```java
  public ObjectName[] getConnectorNames() {
```

```java
  public void destroy() throws LifecycleException {
```

```java
  public void init() {
```

```java
  public ObjectName getObjectName() {
```

```java
  public String getDomain() {
```

No javadoc for public method.

```java
  public int getDebug() {
```

```java
  public String getInfo() {
```

```java
  public String getName() {
```

```java
  public Server getServer() {
```

```java
   public NotificationBroadcasterSupport getBroadcaster() {
```

```java
  public Connector[] findConnectors() {
```

```java
  public String toString() {
```

```java
  public List<LifecycleListener> findLifecycleListeners() {
```

Better to use "@return" javadoc command instead of writing in the description field.

```java
    public Connector findConnector(String name) {
```

Better to use "@return" javadoc command instead of writing in the description field, no "@param" javadoc field for tne "String name" parameter.

- **Checklist C.25**

```java
     private static final Logger log = StandardServer.log;
     private static final ResourceBundle rb =
         log.getResourceBundle();
```

Private static attribute stated before a public static one.

```java
    private String name = null;
```

```java
    private LifecycleSupport lifecycle = new LifecycleSupport(this);
```

```java
    private Server server = null;
```

```java
    private boolean started = false;
```

Private attribute stated before a protected one.

```java
    protected String domain;
    protected ObjectName oname;
```

Protected attribute not in the attributes section.

- **Checklist C.26**

```java
    protected String domain;
    protected ObjectName oname;

    public ObjectName getObjectName() {
        return oname;
    }

    public String getDomain() {
        return domain;
    }
```

These methods and attributes are not grouped by functionality, scope or accessibility.

- **Checklist C.39**

  Note on issue C.39: not everytime a new array is created his elements are initialized using the constructor, but in these cases it's not a problem because the elements are copied from another array that was initialized before.

3.2 Method: addConnector

TODO - inserire commentino inizio sezione

- **Checklist C.33**

```
        Connector results[] = new Connector[connectors.length +
            1];
```

  These declarations are not at the top of the code block.

- **Checklist C.51**

```
        connector.setService(this);
```

  “this” is of the type StandardService, where the parameter of setService is ”Service” (see the Javadoc for setService() of the class Connector).

```
        System.arraycopy(connectors, 0, results, 0,
            connectors.length);
```

  Connectors and result are of the type ”Connector” where the types of the first and the third parameters of System.arraycopy are ”Object” (see the Javadoc for arrayCopy() of the class System).

```
                log.log(Level.SEVERE, "Connector.initialize", e);
```

  The type of “e” is ”LifecycleException” where the type of the third parameter of Logger.log() is ”Throwable” (see the Javadoc for log() of the class Log).

```
                log.log(Level.SEVERE, "Connector.start", e);
```

  The type of “e” is ”LifecycleException” where the type of the third parameter of Logger.log() is ”Throwable” (see the Javadoc for log() of the class Log).

```
        support.firePropertyChange("connector", null,
            connector);
```

Type of the the third parameter is “Object” and ”connector” is of the type ”Connector” (see the Javadoc for firePropertyChange() of the class PropertyChangeSupport).

These lines of code contains implicit types conversions.

3.3 Method: removeConnector

TODO - inserire commentino inizio sezione

- **Checklist C.2**

  In this method it’s declared ”j”, an integer variable that is not used for temporary use. Replace with a significative name.

- **Checklist C.11**

```
      if (j < 0)
        return;
```

```
        if (i != j)
          results[k++] = connectors[i];
```

  These lines of code contains if statements with only one statement to execute and are not surrounded by curly braces.

- **Checklist C.18**

```
  // START SJSAS 6231069
```

```
  // END SJSAS 6231069
```

```
      // START SJSAS 6231069
```

```
      // END SJSAS 6231069
```

```
      // START SJSAS 6231069
```

```
      // END SJSAS 6231069
```

  These comments don’t explain what the code are doing.

- **Checklist C.19**

```java
    //public void removeConnector(Connector connector) {
```

```java
            /*if (started && (connectors[j] instanceof Lifecycle)) {
                try {
                    ((Lifecycle) connectors[j]).stop();
                } catch (LifecycleException e) {
                    log.error("Connector.stop", e);
                }
            }*/
```

```java
            /*connectors[j].setContainer(null);
            connector.setService(null);*/
```

  Commented code does not contain a reason for being commented out.

- **Checklist C.23**

```java
    public void removeConnector(Connector connector) throws
        LifecycleException{
```

  No “@throws” javadoc field for the exception "LifecycleException"

- **Checklist C.33**

```java
            int k = 0;
            Connector results[] = new Connector[connectors.length -
                1];
```

  Declarations are not at the top of the block.

- **Checklist C.40**

```java
                if (connector == connectors[i]) {
```

  Two objects are compared using "==" and not equals().

- **Checklist C.44**

```java
            int j = -1;
            for (int i = 0; i < connectors.length; i++) {
```

```
482                 if (connector == connectors[i]) {
483                     j = i;
484                     break;
485                 }
486             }
```

There is a "break;" into the for() block. Use another iteration block.

```
510                     results[k++] = connectors[i];
```

It's better to explicitly increment 'k' before the assignment with a separed statement.

These lines of code does not avoid "Brutish Programming".

- **Checklist C.51**

```
515             support.firePropertyChange("connector", connector,
                    null);
```

The type of the second parameter has to be "Object" and "connector" is of the type "Connector".

- **Checklist C.56**

```
480             int j = -1;
481             for (int i = 0; i < connectors.length; i++) {
482                 if (connector == connectors[i]) {
483                     j = i;
484                     break;
485                 }
486             }
```

Loops are not correctly formed expecially the termination expression at the line 484 (break).

3.4 Method: start

TODO - inserire commentino inizio sezione

- **Checklist C.11**

```
596         if( ! initialized )
597             init();
```

```
            if (connectors[i] instanceof Lifecycle)
                ((Lifecycle) connectors[i]).start();
```

These lines of code contains if statements with only one statement to execute and are not surrounded by curly braces.

- **Checklist C.51**

```
        log.log(Level.INFO, STARTING_SERVICE, this.name);
```

The type of the third parameter is "Object" where this.name is String.

3.5 Method: stop

TODO - inserire commentino inizio sezione

- **Checklist C.11**

```
            if (connectors[i] instanceof Lifecycle)
                ((Lifecycle) connectors[i]).stop();
```

These lines of code contains if statements with only one statement to execute and are not surrounded by curly braces.

- **Checklist C.51**

```
        log.log(Level.INFO, STOPPING_SERVICE, this.name);
```

Type of the third parameter is "Object" where this.name is String.

3.6 Method: initialize

TODO - inserire commentino inizio sezione

- **Checklist C.13**

```
            String msg =
                MessageFormat.format(rb.getString(ERROR_REGISTER_SERVICE_EXCEPTION),
                domain);
```

Limit of 80 characters exceeded.

```
        String msg =
            MessageFormat.format(rb.getString(FAILED_SERVICE_INIT_EXCEPTION),
            domain);
```

Limit of 80 characters exceeded.

- **Checklist C.18**

```
687         // Service shouldn't be used with embedded, so it doesn't
                matter
```

```
698                 // Hack - Server should be deprecated...
```

```
711             // HACK: ServerFactory should be removed...
```

These three comments don't explain what the code do. These comments are directives to the developers.

- **Checklist C.23**

```
684     public void initialize()
685             throws LifecycleException
```

No "@throws" javadoc field for the exception "LifecycleException"

- **Checklist C.51**

```
703                 String msg =
                        MessageFormat.format(rb.getString(ERROR_REGISTER_SERVICE_EXCEPTION),
                        domain);
```

"domain" is a String and it's implicitly converted into "Object"

```
704                 log.log(Level.SEVERE, msg, e);
```

"e" is an Exception and it's implicitly converted into "Object"

```
713             ServerFactory.getServer().addService(this);
```

"this" is a StandardService and it's implocitly converted into "Service"

- **Checklist C.53**

```
702             } catch (Exception e) {
```

"MalformedObjectNameException" should be caught instead of "Exception"

Chapter 4

Other Problems

```java
public ObjectName[] getConnectorNames() {
    ObjectName results[] = new ObjectName[connectors.length];
    for( int i=0; i<results.length; i++ ) {
        // if it's a coyote connector
        //if( connectors[i] instanceof CoyoteConnector ) {
        //
            results[i]=((CoyoteConnector)connectors[i]).getJmxName();
        //}
    }
    return results;
}
```

This method does nothing, it returns only an empty array.

Chapter 5

Additional Material

Chapter 6

Appendix

6.1 Software Used

6.2 Document References

6.3 Hours of Work