Politecnico di Milano
A.A. 2015-2016
Software Engineering 2 Project

Code Inspection

version 1

Giovanni Brena (858328), Andrea Canale (858638)

Contents

1. Assigned Methods..... 3

2. Functional Analysis..... 3

3. Issues..... 6

4. Code..... 7

1 Assigned Methods

Methods assigned to be inspected are part of **Embedded.java** class file contained into folder:

appserver/web/web-core/src/main/java/org/apache/catalina/startup/

Methods: removeHost(Host host), start(), stop(), initNaming(), initDirs

Analyzed code is available at the end of this document.

2 Functional Analysis

This section provides information about structure and functionality of the assigned methods set. Knowledge on methods mainly comes from **Javadoc**.

Public class **Embedded** allows to embed a Catalina servlet environment inside another application. It contains a set of Engines deployed for this server and provides start, stop and destroy methods. Each Engine must contain at least one Host and each Host has an associated Context object. Connectors objects manage TCP/IP requests.

see: https://tomcat.apache.org/tomcat-5.5-doc/catalina/docs/api/org/apache/catalina/startup/Embedded.html

- **public synchronized void removeHost(Host host)**

  start line: 855
  parameter: Host host
  return value: void
  visibility: public
  used class variables: Logger log, Engine[] engines
  local variables: boolean found, Container[] hosts, int i
  exceptions managed: no
  exceptions thrown: no

functionalities:
Remove the specified Host, along with all of its related Contexts, from the set of defined Hosts for its associated Engine. If this is the last Host for this Engine, the Engine will also be removed.

- public void start()

start line: 958
parameter: no
return value: void
visibility: public
used class variables: Logger log, Engine[] engines, boolean started, Lifecycle lifecycle, ResourceBoundle rb
local variables: int i
exceptions managed: no
exceptions thrown: LifecycleException

functionalities:
Prepare for the beginning of active use of the public methods of this component. This method should be called after configure(), and before any of the public methods of the component are utilized.

- public void stop()

start line: 1009
parameter: no
return value: void
visibility: public
used class variables: Logger log, boolean started, ResourceBoundle rb, Lifecycle lifecycle, Engines[] engines
local variables: int i
exceptions managed: no
exceptions thrown: LifecycleException

functionalities:

Gracefully terminate the active use of the public methods of this component. This method should be the last one called on a given instance of this component.

- protected void initNaming()

  start line: 1058
  parameter: no
  return value: void
  visibility: protected
  used class variables: boolean useNaming, Logger log
  local variables: String value, String oldValue
  exceptions managed: no
  exceptions thrown: no

  functionalities:
  Initialize naming strings if useNaming property is enebled.

- protected void initDirs()

  start line: 1094
  parameter: no
  return value: void
  visibility: protected
  used class variables: no
  local variables: String catalinaHome, String j2eeHome, File home, String catalinaBase, File base
  exceptions managed: IOException
  exceptions thrown: no

  functionalities:
  Initialize directories.

3 Issues

1. public synchronized void removeHost(Host host)

- missing curly braces for if single statement (line 858, 872, 875, 879)
- no control on null parameter host
- comparison through "==" instead of "equals" (line 865)
- line 872: no log printed for not found
- line 879: Type error (space) in log string
- Usage of boolean "found" useless, should move code line 877-880 in place of 866-867 + return

2. public void start()

- line 961: commented out code not explained
- missing curly braces for if single statement (line 966, 979, 989, 995)
- line 966: line break before operator (should be after)
- line 983: boolean variable "initialized" not defined within this class
- line 986: "connectors" set not defined within this class

3. public void stop()

- missing curly braces for if single statement (line 1012, 1016, 1024, 1030)
- line 1022: "connectors" set not defined within this class

4. protected void initNaming()

- line 1052-1056 unclear header comments / javadoc
- line 1062: commented out code not explained
- missing curly braces for "if" single-statement (line 1064, 1074, 1078)
- Hardcoded string parameters should be defined externally (line 1067, 1069, 1070, 1084)
- Line 1087: output error, "alread" instead of "already"

5. protected void initDirs()

- Missing javadoc
- missing header comments
- Hardcoded string parameters should be defined externally (line 1096, 1099, 1101, and more)
- line 1101: duplicate System.getProperty(String) call, suggested catalinaHome=j2eeHome
- line 1103 duplicate System.getProperty(String) call
- System.getProperty("catalina.base") is called 4 times within this method, it should be stored into a unique local variable
- Missing close() statement for files "home" and "base"

4 Code

- removeHost(Host host)

```
848     /**
849      * Remove the specified Host, along with all of its related Contexts,
850      * from the set of defined Hosts for its associated Engine.  If this is
851      * the last Host for this Engine, the Engine will also be removed.
852      *
853      * @param host The Host to be removed
854      */
855     public synchronized void removeHost(Host host) {
856
857         if (log.isLoggable(Level.FINE))
858             log.log(Level.FINE, "Removing host[" + host.getName() + "]");
859
860         // Is this Host actually among those that are defined?
861         boolean found = false;
862         for (int i = 0; i < engines.length; i++) {
863             Container hosts[] = engines[i].findChildren();
864             for (int j = 0; j < hosts.length; j++) {
865                 if (host == (Host) hosts[j]) {
866                     found = true;
867                     break;
868
869                 }
870             }
871             if (found)
872                 break;
873         }
874         if (!found)
875             return;
876
877         // Remove this Host from the associated Engine
878         if (log.isLoggable(Level.FINE))
879             log.log(Level.FINE, " Removing this Host");
880         host.getParent().removeChild(host);
881
882     }
```

- start()

```
949     /**
950      * Prepare for the beginning of active use of the public methods of this
951      * component.  This method should be called after <code>configure()</code>,
952      * and before any of the public methods of the component are utilized.
953      *
954      * @exception LifecycleException if this component detects a fatal error
955      *  that prevents this component from being used
956      */
957     @Override
958     public void start() throws LifecycleException {
959 
960         /* SJSAS 5022949
961         if( log.isInfoEnabled() )
962             log.info("Starting tomcat server");
963          */
964         // START SJSAS 6340446
965         if (log.isLoggable(Level.FINE)) {
966             log.log(Level.FINE, "Starting Servlet container component of "
967                     + ServerInfo.getServerInfo());
968         }
969         // END SJSAS 6340446
970 
971         // Validate the setup of our required system properties
972         initDirs();
973 
974         // Initialize some naming specific properties
975         initNaming();
976 
977         // Validate and update our current component state
978         if (started)
979             throw new LifecycleException
980                 (rb.getString(SERVICE_BEEN_STARTED_EXCEPTION));
981         lifecycle.fireLifecycleEvent(START_EVENT, null);
982         started = true;
983         initialized = true;
984 
985         // Start our defined Connectors first
986         for (int i = 0; i < connectors.length; i++) {
987             connectors[i].initialize();
988             if (connectors[i] instanceof Lifecycle)
989                 ((Lifecycle) connectors[i]).start();
990         }
991 
992         // Start our defined Engines second
993         for (int i = 0; i < engines.length; i++) {
994             if (engines[i] instanceof Lifecycle)
995                 ((Lifecycle) engines[i]).start();
996         }
997     }
```

- stop()

```
1000      /**
1001       * Gracefully terminate the active use of the public methods of this
1002       * component.  This method should be the last one called on a given
1003       * instance of this component.
1004       *
1005       * @exception LifecycleException if this component detects a fatal error
1006       *  that needs to be reported
1007       */
1008      @Override
1009      public void stop() throws LifecycleException {
1010
1011          if (log.isLoggable(Level.FINE))
1012              log.log(Level.FINE, "Stopping embedded server");
1013
1014          // Validate and update our current component state
1015          if (!started)
1016              throw new LifecycleException
1017                  (rb.getString(SERVICE_NOT_BEEN_STARTED_EXCEPTION));
1018          lifecycle.fireLifecycleEvent(STOP_EVENT, null);
1019          started = false;
1020
1021          // Stop our defined Connectors first
1022          for (int i = 0; i < connectors.length; i++) {
1023              if (connectors[i] instanceof Lifecycle)
1024                  ((Lifecycle) connectors[i]).stop();
1025          }
1026
1027          // Stop our defined Engines second
1028          for (int i = 0; i < engines.length; i++) {
1029              if (engines[i] instanceof Lifecycle)
1030                  ((Lifecycle) engines[i]).stop();
1031          }
1032
1033      }
```

- initNaming()

```
1048    /** Initialize naming - this should only enable java:env and root naming.
1049     * If tomcat is embeded in an application that already defines those -
1050     * it shouldn't do it.
1051     *
1052     * XXX The 2 should be separated, you may want to enable java: but not
1053     * the initial context and the reverse
1054     * XXX Can we "guess" - i.e. lookup java: and if something is returned assume
1055     * false ?
1056     * XXX We have a major problem with the current setting for java: url
1057     */
1058    protected void initNaming() {
1059        // Setting additional variables
1060        if (!useNaming) {
1061            // START SJSAS 5031700
1062            //log.info( "Catalina naming disabled");
1063            if (log.isLoggable(Level.FINE)) {
1064                log.log(Level.FINE, "Catalina naming disabled");
1065            }
1066            // END SJSAS 5031700
1067            System.setProperty("catalina.useNaming", "false");
1068        } else {
1069            System.setProperty("catalina.useNaming", "true");
1070            String value = "org.apache.naming";
1071            String oldValue =
1072                System.getProperty(javax.naming.Context.URL_PKG_PREFIXES);
1073            if (oldValue != null) {
1074                value = value + ":" + oldValue;
1075            }
1076            System.setProperty(javax.naming.Context.URL_PKG_PREFIXES, value);
1077            if (log.isLoggable(Level.FINE))
1078                log.log(Level.FINE, "Setting naming prefix=" + value);
1079            value = System.getProperty
1080                (javax.naming.Context.INITIAL_CONTEXT_FACTORY);
1081            if (value == null) {
1082                System.setProperty
1083                    (javax.naming.Context.INITIAL_CONTEXT_FACTORY,
1084                     "org.apache.naming.java.javaURLContextFactory");
1085            } else {
1086                if (log.isLoggable(Level.FINE)) {
1087                    log.log(Level.FINE, "INITIAL_CONTEXT_FACTORY alread set " + value);
1088                }
1089            }
1090        }
1091    }
```

- initDirs()

```
1094    protected void initDirs() {
1095
1096        String catalinaHome = System.getProperty("catalina.home");
1097        if (catalinaHome == null) {
1098            // Backwards compatibility patch for J2EE RI 1.3
1099            String j2eeHome = System.getProperty("com.sun.enterprise.home");
1100            if (j2eeHome != null) {
1101                catalinaHome=System.getProperty("com.sun.enterprise.home");
1102            } else if (System.getProperty("catalina.base") != null) {
1103                catalinaHome = System.getProperty("catalina.base");
1104            } else {
1105                // Use IntrospectionUtils and guess the dir
1106                catalinaHome = IntrospectionUtils.guessInstall
1107                    ("catalina.home", "catalina.base", "catalina.jar");
1108                if (catalinaHome == null) {
1109                    catalinaHome = IntrospectionUtils.guessInstall
1110                        ("tomcat.install", "catalina.home", "tomcat.jar");
1111                }
1112            }
1113        }
1114        // last resort - for minimal/embedded cases.
1115        if(catalinaHome==null) {
1116            catalinaHome=System.getProperty("user.dir");
1117        }
1118        if (catalinaHome != null) {
1119            File home = new File(catalinaHome);
1120            if (!home.isAbsolute()) {
1121                try {
1122                    catalinaHome = home.getCanonicalPath();
1123                } catch (IOException e) {
1124                    catalinaHome = home.getAbsolutePath();
1125                }
1126            }
1127            System.setProperty("catalina.home", catalinaHome);
1128        }
1129
1130        if (System.getProperty("catalina.base") == null) {
1131            System.setProperty("catalina.base",
1132                               catalinaHome);
1133        } else {
1134            String catalinaBase = System.getProperty("catalina.base");
1135            File base = new File(catalinaBase);
1136            if (!base.isAbsolute()) {
1137                try {
1138                    catalinaBase = base.getCanonicalPath();
1139                } catch (IOException e) {
1140                    catalinaBase = base.getAbsolutePath();
1141                }
1142            }
1143            System.setProperty("catalina.base", catalinaBase);
1144        }
1145
1146    }
```