ABOUT SATYA CHOUDHURY

Satya Choudhury is a IT professional with over 23+ years of experience in multiple technologies such as Java, IBM AS/400 (iSeries) and Web (PHP, Vuejs, Codeigniter, Bootstrap, etc.). Apart from programming he also possess excellent UI/UX design skills and runs his own store at http://satyatunes.com.

JBoss WildfFly Logging Configuration Example

Posted by: Satya Choudhury in JBoss WildFly September 25th, 2018 1 Comment 4409 Views

1. Introduction

In this example we will review the

```
wildFly
```

```
WildFly
```

is an application server written in

```
Java
```

that implements

```
Java EE
```

specification. It's a fast and lightweight server. It's built on a modular service container that enables services on demand when needed by the application. The latest release as of this this writing is 14.0.1, which is

```
Java EE 8
```

certified.

1.2 WildFly Logging Configuration

```
Logging
```

subsystem represents the overall server

```
logging
```

configuration. It is made up of the following four parts:

- Handler
- Logger
- Root Logger
- Logging Profiles

1.2.1 Handlers

:

- async-handler – An async-handler is a `handler` that asynchronously writes log messages to it's child `handlers`. This type of `handler` is generally used to wrap other `handlers` that take a substantial time to write messages.
- console-handler – A console-handler is a `handler` that writes log messages to the `console`. Generally this writes to `stdout`, but can be set to write to `stderr`.
- custom-handler – A custom-handler allows you to define any

```
async-handler
```

.

- file-handler – A file-handler is a

```
handler
```

that writes log messages to the specified file.

- periodic-rotating-file-handler – A periodic-rotating-file-handler is a

```
handler
```

that writes log messages to the specified file. The file rotates on the date pattern specified in the suffix attribute. The suffix must be a valid pattern recognized by the

```
java.text.SimpleDateFormat
```

and must not rotate on seconds or milliseconds.

- periodic-size-rotating-file-handler – A periodic-size-rotating-file-handler is a

```
handler
```

that writes log messages to the specified file. The file rotates on the date pattern specified in the suffix attribute or the rotate-size attribute. The suffix must be a valid pattern recognized by the java.text.SimpleDateFormat and must not rotate on seconds or milliseconds.

- size-rotating-file-handler – A size-rotating-file-handler is a

```
handler
```

that writes log messages to the specified file. The file rotates when the file size is greater than the rotate-size attribute. The rotated file will be kept and the index appended to the name moving previously rotated file indexes up by 1 until the max-backup-index is reached. Once the max-backup-index is reached, the indexed files will be overwritten.

- socket-handler – A socket-handler is a

```
handler
```

which sends messages over a

and must be defined in a

```
socket
```

binding group under the

```
local-destination-outbound-socket-binding
```

or

```
remote-destination-outbound-socket-binding
```

resource.

- syslog-handler – A syslog-handler is a

```
handler
```

that writes to a

```
syslog
```

server via

```
UDP
```

. The

```
handler
```

support

```
RFC3164
```

or

```
RFC5424
```

formats

A

```
logger
```

is the first step to determining if a messages should be logged or not. If a

```
logger
```

is defined with a level, the level of the message must be greater than the level defined on the

```
logger
```

. The filter is then checked next and the rules of the filter will determine whether or not the messages is said to be loggable.

A

```
logger
```

has the following attributes:

- filter-spec – The filter-spec attribute is an expression based string to define filters for the

  ```
  logger
  ```

  .
- #handlers – The

  ```
  handlers
  ```

  attribute is a list of

  ```
  handler
  ```

  names that should be attached to the logger. If the

  ```
  use-parent-handlers
  ```

  attribute is set to true and the log messages is determined to be loggable, parent

  ```
  loggers
  ```

attribute is a

```
boolean
```

attribute to determine whether or not parent

```
loggers
```

should also process the log message.

1.2.3 Root Logger

The root

```
logger
```

defines the level of messages to be logged. You can use this to limit the logging. It reference a

```
handler
```

or set of

```
handlers
```

. Each

```
handler
```

in turn declares the log format and output:

1.2.4 Logging Profiles

```
Logging
```

profiles are like additional

```
logging
```

`logger`

and the

`root logger`

declarations.

You can assign a

`logging`

profile to a deployment via the deployments manifest. Add a Logging-Profile entry to the

`MANIFEST.MF`

file with a value of the

`logging`

profile id. For example a logging profile defined on

`/subsystem=logging/logging-profile=demo`

the

`MANIFEST.MF`

would look like:

```
Manifest-Version: 1.0
Logging-Profile: demo
```

One

`logging`

profile can be assigned to multiple deployments. Using a

`logging`

1.3 Logging Formatter

`Logging`

formatter is used to format the log messages. A formatter can be assigned to a

`handler`

.

`WildFly`

`logging`

subsystem includes following type of formatters:

- JSON Formatter – It's used to format log messages in
  `JSON`
- Pattern Formatter – It's used to format log messages in plain text.
- XML Formatter – It's used to format log messages in
  `XML`
- Custom Formatter – Note that most log records are formatted in the
  `printf`

  format.

2. WildFly Logging Configuration Example

Now it's time to apply what we have learnt so far in a real world example. We will configure logging for our

`Web Application`

```
NetBeans
```

project to follow along this example.

2.1 Technologies used

For this example, we will use the following tools in a

```
Windows
```

```
64-bit
```

platform:

- NetBeans – 8.2
- Java – 1.8.0_161
- WildFly – 14.0.1

2.2 Logging Configuration

The default log files for a standalone server can be found in the log subdirectory. The folder path is

```
./standalone/log/server.log
```

. The configuration files are in

```
XML
```

format and are available in

```
./standalone/configuration
```

. As we are using the full standalone version of the server we are interested in

```
standalone-full.xml
```

```
02    <formatter>
03      <named-formatter name="PATTERN"/>
04    </formatter>
05    <file relative-to="jboss.server.log.dir" path="jboss-wildfly-netbeans-example.log"/>
06    <suffix value=".yyyy-MM-dd"/>
07    <append value="true"/>
08  </periodic-rotating-file-handler>
09  <logger category="com.jcg" use-parent-handlers="false">
10    <level name="INFO"/>
11    <handlers>
12      <handler name="MY_HANDLER"/>
13    </handlers>
14  </logger>
```

- Line 1: We are adding a

  `handler`

  ;

  `periodic-rotating-file-handler`

  to be specific with name

  `MY_HANDLER`

- Line 2 – 4: We are using the

  `formatter`

  called

  `PATTERN`

  . It’s already defined in the configuration file. We are simply using it.
- Line 5: This is where we specify the location of our log file. In this case, our log file will be placed in whichever folder is defined for the server log files.
- Line 6: The date will be suffixed to the file when it’s rotated
- Line 7: Flag to indicate that the date will be appended to the file name
- Line 9:

. In this case, the `logger` `com.jcg` is the parent logger of `com.jcg.wildflyexample`

- Line 10: The level attribute allows the minimum level to allow messages to be logged at for the logger. In this case we are logging anything above `INFO` level
- Line 11 – 13: Making sure the `logger` uses our handler called `MY_HANDLER`

That's all we need for now. Save the file.

2.3 Java Code Change

Let's modify our `Java` `Bean`

```
04 import javax.enterprise.context.RequestScoped;
05 import org.jboss.logging.Logger;
06
07 /**
08  *
09  * @author Satya Choudhury
10  */
11 @Named(value = "greetingsBean")
12 @RequestScoped
13 public class GreetingsBean {
14
15     private String userName = "";
16     private static Logger log = Logger.getLogger(GreetingsBean.class.getName());
17
18     /**
19      * Creates a new instance of GreetingsBean
20      */
21     public GreetingsBean() {
22         //System.out.println("Created GreetingsBean instance...");
23         log.info("Created GreetingsBean instance...");
24     }
25
26     public String getUserName() {
27         return this.userName.trim();
28     }
29
30     public void setUserName(String userName) {
31         this.userName = userName.trim();
32     }
33
34     public String greetUser() {
35         return "greeting";
36     }
37 }
```

- Line 5: We imported the

```
WildFly
```

```
logger
```

```
package
```

```
logger
```

- Line 23: We logged an informational message. There are methods available for

```
debug
```

,

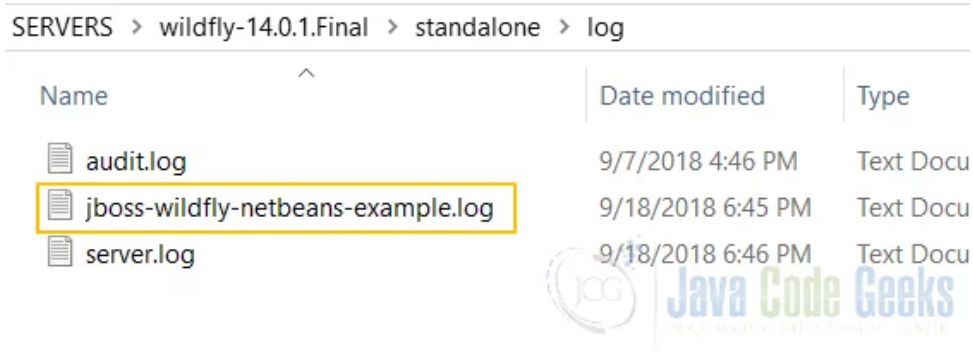
```
warn
```

, etc. I will encourage you to read the documentation.

We are ready to see the configuration in action so, save the file and run the application.

```
NetBeans
```

will start the server and deploy the application. On the welcome page of our application, enter a name then press submit. Verify that the new log file is generated and log messages are written correctly. To do so, navigate to the log folder of the server. You should see the log file. In my case the log file is named **jboss-wildfly-netbeans-example.log** file.



WildFly log file

```
2018-09-18 18:45:02,311 INFO  [com.jcg.wildflyexample.GreetingsBean]
(default task-1) Created GreetingsBean instance...
```


WildFly log file contents

3. JBoss WildfFly Logging Configuration – Summary

In this example, we reviewed the various attributes and options available for logging configuration of

`WildFly`

server. We applied the configuration to our web application to generate separate log files instead of writing the log messages to default server log file.

4. Download the Source Code

This was an example of

`JBoss`

`WildFly`

logging configuration.

Download
You can download the full source code of this example: **Download the NetBeans project**

Tagged with: JBOSS LOGGING WILDFLY

+ 0 — Reply

KNOWLEDGE BASE

- Courses
- Minibooks
- News
- Resources
- Tutorials

THE CODE GEEKS NETWORK

- .NET Code Geeks
- Java Code Geeks
- System Code Geeks
- Web Code Geeks

HALL OF FAME

- Android Alert Dialog Example
- Android OnClickListener Example
- How to convert Character to String and a String to Character Array in Java
- Java Inheritance example
- Java write to File Example
- java.io.FileNotFoundException – How to solve File Not Found Exception
- java.lang.arrayindexoutofboundsexception – How to handle Array Index Out Of Bounds Exception
- java.lang.NoClassDefFoundError – How to solve No Class Def Found Error
- JSON Example With Jersey + Jackson
- Spring JdbcTemplate Example

ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on creating the ultimate Java to Java developers resource center; targeted at the technical architect, technical team lead (senior developer), project manager and junior developers alike. JCGs serve the Java, SOA, Agile and Telecom communities with daily news written by domain experts, articles, tutorials, reviews, announcements, code snippets and open source projects.

DISCLAIMER