**Логирование с Slf4j и Logback**

Java существует уже много лет. В этом языке программирования огромное количество древнего наследия, особенностей и запутывающих новичков костылей. В первых версиях Java не было нормальной системы логирования, что привело к настоящему кошмару и порождению целого моря несовместимых друг с другом стандартов и библиотек, решающих одну и ту же задачу.

Картинка с https://xkcd.ru/927/, наглядно демонстрирующая путь Java в части логирования

В прошлых статьях я описывал кучу различных библиотек логирования: [System.err](https://urvanov.ru/2019/07/03/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d0%b2-java-%d1%81-system-err/), [JUL](https://urvanov.ru/2019/07/03/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-jul-java-util-logging/), [Log4j 1.2](https://urvanov.ru/2019/07/04/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-log4j-1-2/), [Apache Commons Logging](https://urvanov.ru/2019/07/04/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-apache-commons-logging/), [Log4j 2](https://urvanov.ru/2019/07/04/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-log4j-2-%d0%b2-java/). В новых приложениях, как правило, ни один из них не используется. Сейчас правильным подходом считается использование API Slf4j и его реализации Logback.

Но что делать со всем старым кодом? Мы же не можем просто выбросить то огромное количество логеров и библиотек, которое уже существует. Для них нужно подключать специальные зависимости, содержащие их API, но вместо реализации перенаправляющие вывод в Slf4j:

* jcl-over-slf4j.jar содержит в себе API от [Apache Commons Logging](https://urvanov.ru/2019/07/04/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-apache-commons-logging/), но вместо его реализации просто перенаправляет все вызовы в Slf4j.
* log4j-over-slf4j.jar содержит в себе API от Log4j, но вместо его реализации перенаправляет все вызовы в Slf4j.
* jul-to-slf4j.jar содержит в себе обработчик (Handler) для [JUL](https://urvanov.ru/2019/07/03/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-jul-java-util-logging/), который пишет все сообщения в Slf4j. Так как JUL встроен в JDK, то заменить его как в случае Apache Commons Logging и Log4j мы не можем, именно поэтому мы просто добавляем новый Handler.

Кроме вышеперечисленных зависимостей, перенаправляющих в Slf4j с API других библиотек, существуют зависимости, которые наоборот реализуют API Slf4j:

* slf4j-log4j12.jar перенаправляет вызовы Slf4j в Log4j12, то есть позволяет использовать [Log4j 1.2](https://urvanov.ru/2019/07/04/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-log4j-1-2/) в качестве реализации API Slf4.
* slf4j-jdk14.jar перенаправляет вызовы Slf4j в JUL, то есть позволяет использовать [JUL](https://urvanov.ru/2019/07/03/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-jul-java-util-logging/) в качестве реализации API Slf4j.
* slf4j-nop.jar просто игнорирует все вызовы Slf4j, что равносильно полному отключению логов.
* slf4j-simple.jar перенаправляет вызовы Slf4j в [System.err](https://urvanov.ru/2019/07/03/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d0%b2-java-%d1%81-system-err/).
* slf4j-jcl.jar перенаправляет вызовы Slf4j в Apache Commons Logging, то есть позволяет использовать Apache Commons Logging в качестве реализации API Slf4j. Самое интересное в этом случае то, что [Apache Commons Logging](https://urvanov.ru/2019/07/04/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-apache-commons-logging/) тоже является лишь обёрткой с API, перенаправляющей выводы в другие реализации…
* logback-classic.jar — это библиотека логирования, напрямую реализующая API Slf4j. В современных приложениях, как правило, используют именно её.

Надеюсь, я вас не запутал. Итак, что нам нужно сделать, чтобы использовать связку Slf4j и Logback:

1. Подключить slf4j-api.
2. Подключить logback-classic.
3. Подключить jcl-over-slf4j, log4j-over-slf4j, чтобы сообщения логов от зависимостей, которые используют Apache Commons Logging и Log4j перенаправлялись в Slf4j. Можно ещё подключить jul-to-slf4j, но это не рекомендуется, так как от него сильно падает производительность.
4. Из всех других подключаемых зависимостей убирать с помощью exclude в Maven зависимость от конкретной библиотеки логирования.
5. Настроить Logback.
6. Использовать slf4j-api для записи логов.

Давайте сделаем простое приложение с использованием Slf4j. Создайте новый проект Maven. Добавьте туда зависимость от Logback и Slf4j-api:

pom.xml

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | <project xmlns="http://maven.apache.org/POM/4.0.0"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">      <modelVersion>4.0.0</modelVersion>        <groupId>ru.urvanov.javaexamples</groupId>      <artifactId>slf4j-logback-example</artifactId>      <version>0.0.1-SNAPSHOT</version>      <packaging>jar</packaging>        <name>slf4j-logback-example</name>      <url>http://maven.apache.org</url>        <properties>          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>          <java-version>1.8</java-version>          <slf4j.version>1.7.26</slf4j.version>      </properties>        <dependencies>          <dependency>              <groupId>junit</groupId>              <artifactId>junit</artifactId>              <version>4.12</version>              <scope>test</scope>          </dependency>          <dependency>              <groupId>ch.qos.logback</groupId>              <artifactId>logback-classic</artifactId>              <version>1.2.3</version>          </dependency>          <dependency>              <groupId>org.slf4j</groupId>              <artifactId>slf4j-api</artifactId>              <version>${slf4j.version}</version>          </dependency>      </dependencies>      <build>          <plugins>              <plugin>                  <groupId>org.apache.maven.plugins</groupId>                  <artifactId>maven-compiler-plugin</artifactId>                  <version>2.5.1</version>                  <configuration>                      <source>${java-version}</source>                      <target>${java-version}</target>                      <encoding>${project.build.sourceEncoding}</encoding>                      <compilerArgument>-Xlint:all</compilerArgument>                      <showWarnings>true</showWarnings>                      <showDeprecation>true</showDeprecation>                  </configuration>              </plugin>          </plugins>      </build>    </project> |

Logback сначала пытается читать конфигурацию из “logback-test.xml” в classpath, затем из “logback.groovy” в classpath, а затем из “logback.xml” в classpath.

Файл “logback-test.xml” обычно создают в “src/test/resources”, чтобы иметь отдельную конфигурацию для тестов, а файл “logback.xml” обычно создают в “src/main/resources” как основную конфигурацию логирования. Создадим простой файл “logback.xml”:

logback.xml

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | {{urvanov-syntax-highlighter-internal:0}}  <configuration>    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">      <!-- encoders are assigned the type           ch.qos.logback.classic.encoder.PatternLayoutEncoder by default -->      <encoder>        <pattern>%d{dd.MM.yyyy HH:mm:ss.SSS} [%thread] %-5level %logger{20} - %msg%n</pattern>      </encoder>    </appender>      <root level="debug">      <appender-ref ref="STDOUT" />    </root>  </configuration> |

Мы просто указываем уровень debug для корневого логера, а затем подключаем к нему appender, который будет писать в консоль. В pattern мы указываем, что в лог нужно писать сначала дату с временем %d{dd.MM.yyyy HH:mm:ss.SSS}, потом название потока [%thread], затем пять символов уровеня логирования %-5level, затем название логера, пытаясь уместить его в 36 символов, %logger{36}, затем сообщение логера %msg и перевод строки %n.

Напишем простой класс, использующий связку Slf4j и Logback:

Slf4jLogbackExampleApp.java

Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | package ru.urvanov.javaexamples.slf4jlogback;    import java.io.IOException;  import java.nio.file.Files;  import java.nio.file.Paths;    import org.slf4j.Logger;  import org.slf4j.LoggerFactory;    public class Slf4jLogbackExampleApp {      private static final Logger logger = LoggerFactory.getLogger(              Slf4jLogbackExampleApp.class);        private static final String FILENAME = "/file/does/not/exist";        public static void main(String[] args) {          logger.info("Just a log message.");          logger.debug("Message for debug level.");          try {              Files.readAllBytes(Paths.get(FILENAME));          } catch (IOException ioex) {              logger.error("Failed to read file {}.", FILENAME, ioex);          }      }  } |

Обратите внимание, что мы используем классы из Slf4j, а не из библиотеки Logback. Сначала мы получаем логгер:

Java

|  |  |
| --- | --- |
| 1  2 | private static final Logger logger = LoggerFactory.getLogger(          Slf4jLogbackExampleApp.class); |

Затем выводим два тестовых сообщения в лог:

Java

|  |  |
| --- | --- |
| 1  2 | logger.info("Just a log message.");  logger.debug("Message for debug level."); |

А дальше воспользуемся достижением Slf4j: мы выводем в лог строку с параметром:

Java

|  |  |
| --- | --- |
| 1 | logger.error("Failed to read file {}.", FILENAME, ioex); |

Вместо {} в выходную строку будет подставляться FILENAME. Раньше, [например в Apache Commons Logging](https://urvanov.ru/2019/07/04/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-apache-commons-logging/), мы использовали String.format, но это приводит к дополнительным расходам, так как String.format вычисляется даже в том случае, когда в реальности строка в лог не попадёт из-за выбранного уровня логирования. Например, если мы пишем сообщение с уровнем DEBUG, а в конфигурации настроено, что в лог нужно выводить только INFO.

В случае же Slf4j мы указываем {} в тех местах, куда нужно подставить параметры при выводе лога, а затем передаём нужные значение в последующих параметрах метода. Последним параметром идёт, как правило, само исключение ( ioex).

Теперь немного модифицируем файл конфигурации, чтобы добавить вывод сообщений лога в файл:

logback.xml

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | {{urvanov-syntax-highlighter-internal:0}}  <configuration>      <appender name="STDOUT"          class="ch.qos.logback.core.ConsoleAppender">          <!-- encoders are assigned the type ch.qos.logback.classic.encoder.PatternLayoutEncoder              by default -->          <encoder>              <pattern>%d{dd.MM.yyyy HH:mm:ss.SSS} [%thread] %-5level  %logger{20} - %msg%n</pattern>          </encoder>      </appender>        <appender name="FILE"          class="ch.qos.logback.core.rolling.RollingFileAppender">          <file>slf4jlogbackexample.log</file>          <rollingPolicy              class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">              <!-- daily rollover -->              <fileNamePattern>slf4jlogbackexample.%d{yyyy-MM-dd}.log</fileNamePattern>                <!-- Храним файлы логов 10 дней -->              <maxHistory>10</maxHistory>                <!-- Максимальный размер файлов лога 30 гигабайт -->              <totalSizeCap>10GB</totalSizeCap>            </rollingPolicy>          <encoder>              <pattern>%d{dd.MM.yyyy HH:mm:ss.SSS} [%thread] %-5level  %logger{20} - %msg%n</pattern>          </encoder>      </appender>        <root level="debug">          <appender-ref ref="STDOUT" />          <appender-ref ref="FILE" />      </root>  </configuration> |

Мы добавили ещё один appender с именем FILE, в качестве реализации appender-а выбрали RollingFileAppender и настроили его на то, чтобы он создавал новый файл каждый день, максимально хранил логи 10 дней и до 30 гигабайт.

Если мы сейчас запустим наш класс Slf4jLogbackExampleApp, то в консоли и в файле “slf4jlogbackexample.log” увидим следующее содержимое:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | 08.07.2019 11:03:27.083 [main] INFO   r.u.j.s.Slf4jLogbackExampleApp - Just a log message.  08.07.2019 11:03:27.093 [main] DEBUG  r.u.j.s.Slf4jLogbackExampleApp - Message for debug level.  08.07.2019 11:03:27.111 [main] ERROR  r.u.j.s.Slf4jLogbackExampleApp - Failed to read file /file/does/not/exist.  java.nio.file.NoSuchFileException: \file\does\not\exist  at sun.nio.fs.WindowsException.translateToIOException(WindowsException.java:79)  at sun.nio.fs.WindowsException.rethrowAsIOException(WindowsException.java:97)  at sun.nio.fs.WindowsException.rethrowAsIOException(WindowsException.java:102)  at sun.nio.fs.WindowsFileSystemProvider.newByteChannel(WindowsFileSystemProvider.java:230)  at java.nio.file.Files.newByteChannel(Files.java:361)  at java.nio.file.Files.newByteChannel(Files.java:407)  at java.nio.file.Files.readAllBytes(Files.java:3152)  at ru.urvanov.javaexamples.slf4jlogback.Slf4jLogbackExampleApp.main(Slf4jLogbackExampleApp.java:20) |

Отлично. Логирование работает. Теперь потренируемся в настраивании bridge-ей, который будут перенаправлять логи из других логеров в slf4j. Для этого нам нужно подключить какую-нибудь внешнюю библиотеку, которая вместо slf4j использует в качестве логирования что-то другое. Я сходу не смог найти что-то подобное, поэтому создал сам. Создайте новый проект Maven с “pom.xml”:

pom.xml

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | <project xmlns="http://maven.apache.org/POM/4.0.0"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">      <modelVersion>4.0.0</modelVersion>        <groupId>ru.urvanov.javaexamples</groupId>      <artifactId>slf4jlogback-commons-logging-example</artifactId>      <version>0.0.1-SNAPSHOT</version>      <packaging>jar</packaging>        <name>slf4jlogback-commons-logging-example</name>      <url>http://maven.apache.org</url>        <properties>          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>          <java-version>1.8</java-version>      </properties>        <dependencies>          <dependency>              <groupId>commons-logging</groupId>              <artifactId>commons-logging</artifactId>              <version>1.2</version>          </dependency>      </dependencies>      <build>          <plugins>              <plugin>                  <groupId>org.apache.maven.plugins</groupId>                  <artifactId>maven-compiler-plugin</artifactId>                  <version>2.5.1</version>                  <configuration>                      <source>${java-version}</source>                      <target>${java-version}</target>                      <encoding>${project.build.sourceEncoding}</encoding>                      <compilerArgument>-Xlint:all</compilerArgument>                      <showWarnings>true</showWarnings>                      <showDeprecation>true</showDeprecation>                  </configuration>              </plugin>          </plugins>      </build>  </project> |

Затем создайте класс “Slf4jLogbackCommonsLoggingExample”:

Slf4jLogbackCommonsLoggingExample.java

Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package ru.urvanov.javaexamples.slf4jlogbackcommonslogging;    import org.apache.commons.logging.Log;  import org.apache.commons.logging.LogFactory;    public class Slf4jLogbackCommonsLoggingExample {      private static final Log log = LogFactory.getLog(              Slf4jLogbackCommonsLoggingExample.class);        public static void logCommonsLogging() {          log.debug("Debug message for commons logging");          log.info("info message to commons logging");      }  } |

Мы просто пишем пару сообщений в лог. Соберём этот модуль и положим его в локальный репозиторий Maven:

|  |  |
| --- | --- |
| 1 | mvn clean install |

Теперь в наш основной проект slf4j-logback-example добавим новую зависимость:

pom.xml

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>ru.urvanov.javaexamples</groupId>      <artifactId>slf4jlogback-commons-logging-example</artifactId>      <version>0.0.1-SNAPSHOT</version>  </dependency> |

Подключая её мы заодно получаем Apache Commons Logging в нашем classpath, что можно легко увидеть в дереве зависимостей:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | $ mvn dependency:tree    [INFO] Scanning for projects...  [INFO]  [INFO] -----------< ru.urvanov.javaexamples:slf4j-logback-example >------------  [INFO] Building slf4j-logback-example 0.0.1-SNAPSHOT  [INFO] --------------------------------[ jar ]---------------------------------  [INFO]  [INFO] --- maven-dependency-plugin:2.8:tree (default-cli) @ slf4j-logback-example ---  [INFO] ru.urvanov.javaexamples:slf4j-logback-example:jar:0.0.1-SNAPSHOT  [INFO] +- junit:junit:jar:4.12:test  [INFO] |  \- org.hamcrest:hamcrest-core:jar:1.3:test  [INFO] +- ch.qos.logback:logback-classic:jar:1.2.3:compile  [INFO] |  \- ch.qos.logback:logback-core:jar:1.2.3:compile  [INFO] +- org.slf4j:slf4j-api:jar:1.7.26:compile  [INFO] \- ru.urvanov.javaexamples:slf4jlogback-commons-logging-example:jar:0.0.1-SNAPSHOT:compile  [INFO]    \- commons-logging:commons-logging:jar:1.2:compile  [INFO] ------------------------------------------------------------------------  [INFO] BUILD SUCCESS  [INFO] ------------------------------------------------------------------------  [INFO] Total time:  1.760 s  [INFO] Finished at: 2019-07-08T11:32:44+03:00  [INFO] ------------------------------------------------------------------------ |

В листинге выше на 17 строке видно, что заодно мы получили зависимость от commons-logging. Нам нужно её убрать. В нашем “pom.xml” меняем подключение зависимости slf4jlogback-commons-logging-example вот так:

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <dependency>      <groupId>ru.urvanov.javaexamples</groupId>      <artifactId>slf4jlogback-commons-logging-example</artifactId>      <version>0.0.1-SNAPSHOT</version>      <exclusions>          <exclusion>              <groupId>commons-logging</groupId>              <artifactId>commons-logging</artifactId>          </exclusion>      </exclusions>  </dependency> |

Посмотрим дерево зависимостей снова:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | $ mvn dependency:tree    [INFO] Scanning for projects...  [INFO]  [INFO] -----------< ru.urvanov.javaexamples:slf4j-logback-example >------------  [INFO] Building slf4j-logback-example 0.0.1-SNAPSHOT  [INFO] --------------------------------[ jar ]---------------------------------  [INFO]  [INFO] --- maven-dependency-plugin:2.8:tree (default-cli) @ slf4j-logback-example ---  [INFO] ru.urvanov.javaexamples:slf4j-logback-example:jar:0.0.1-SNAPSHOT  [INFO] +- junit:junit:jar:4.12:test  [INFO] |  \- org.hamcrest:hamcrest-core:jar:1.3:test  [INFO] +- ch.qos.logback:logback-classic:jar:1.2.3:compile  [INFO] |  \- ch.qos.logback:logback-core:jar:1.2.3:compile  [INFO] +- org.slf4j:slf4j-api:jar:1.7.26:compile  [INFO] \- ru.urvanov.javaexamples:slf4jlogback-commons-logging-example:jar:0.0.1-SNAPSHOT:compile  [INFO] ------------------------------------------------------------------------  [INFO] BUILD SUCCESS  [INFO] ------------------------------------------------------------------------  [INFO] Total time:  1.637 s  [INFO] Finished at: 2019-07-08T11:36:55+03:00  [INFO] ------------------------------------------------------------------------ |

Уже лучше. Но раз мы убрали Apache Commons Logging, то наш slf4jlogback-commons-logging-example не сможет найти классы org.apache.commons.logging.Log и org.apache.commons.logging.LogFactory. Что же нам делать? Мы должны подключить зависимость jcl-over-slf4j, она содержит нужные классы, но вместо настоящей реализации Apache Commons Logging просто перенаправляет вызовы в Slf4j:

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>org.slf4j</groupId>      <artifactId>jcl-over-slf4j</artifactId>      <version>${slf4j.version}</version>  </dependency> |

Теперь мы можем в основном классе Slf4jLogbackExampleApp вызвать наш метод, а в логе увидеть его сообщения:

Slf4jLogbackExampleApp.java

Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | package ru.urvanov.javaexamples.slf4jlogback;    import java.io.IOException;  import java.nio.file.Files;  import java.nio.file.Paths;    import org.slf4j.Logger;  import org.slf4j.LoggerFactory;    import ru.urvanov.javaexamples.slf4jlogbackcommonslogging.Slf4jLogbackCommonsLoggingExample;    public class Slf4jLogbackExampleApp {      private static final Logger logger = LoggerFactory.getLogger(              Slf4jLogbackExampleApp.class);        private static final String FILENAME = "/file/does/not/exist";        public static void main(String[] args) {          logger.info("Just a log message.");          logger.debug("Message for debug level.");          try {              Files.readAllBytes(Paths.get(FILENAME));          } catch (IOException ioex) {              logger.error("Failed to read file {}.", FILENAME, ioex);          }          Slf4jLogbackCommonsLoggingExample.logCommonsLogging();      }  } |

Если мы запустим это на выполнение, то в консоли и в файле “slf4jlogbackexample.log” мы увидим следующие строки:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | 08.07.2019 11:44:52.227 [main] INFO   r.u.j.s.Slf4jLogbackExampleApp - Just a log message.  08.07.2019 11:44:52.227 [main] DEBUG  r.u.j.s.Slf4jLogbackExampleApp - Message for debug level.  08.07.2019 11:44:52.249 [main] ERROR  r.u.j.s.Slf4jLogbackExampleApp - Failed to read file /file/does/not/exist.  java.nio.file.NoSuchFileException: \file\does\not\exist  at sun.nio.fs.WindowsException.translateToIOException(WindowsException.java:79)  at sun.nio.fs.WindowsException.rethrowAsIOException(WindowsException.java:97)  at sun.nio.fs.WindowsException.rethrowAsIOException(WindowsException.java:102)  at sun.nio.fs.WindowsFileSystemProvider.newByteChannel(WindowsFileSystemProvider.java:230)  at java.nio.file.Files.newByteChannel(Files.java:361)  at java.nio.file.Files.newByteChannel(Files.java:407)  at java.nio.file.Files.readAllBytes(Files.java:3152)  at ru.urvanov.javaexamples.slf4jlogback.Slf4jLogbackExampleApp.main(Slf4jLogbackExampleApp.java:22)  08.07.2019 11:44:52.249 [main] DEBUG  r.u.j.s.Slf4jLogbackCommonsLoggingExample - Debug message for commons logging  08.07.2019 11:44:52.249 [main] INFO   r.u.j.s.Slf4jLogbackCommonsLoggingExample - info message to commons logging |

С Apache Commons Logging разобрались. Теперь разберём ситуацию, когда мы подключаем внешнюю библиотеку, использующую Log4j. Создадим эту библиотеку вручную. Файл “pom.xml”:

pom.xml

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | <project xmlns="http://maven.apache.org/POM/4.0.0"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">      <modelVersion>4.0.0</modelVersion>        <groupId>ru.urvanov.javaexamples</groupId>      <artifactId>slf4jlogback-log4j-example</artifactId>      <version>0.0.1-SNAPSHOT</version>      <packaging>jar</packaging>        <name>slf4jlogback-log4j-example</name>      <url>http://maven.apache.org</url>        <properties>          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>          <java-version>1.8</java-version>      </properties>        <dependencies>          <dependency>              <groupId>log4j</groupId>              <artifactId>log4j</artifactId>              <version>1.2.17</version>          </dependency>      </dependencies>      <build>          <plugins>              <plugin>                  <groupId>org.apache.maven.plugins</groupId>                  <artifactId>maven-compiler-plugin</artifactId>                  <version>2.5.1</version>                  <configuration>                      <source>${java-version}</source>                      <target>${java-version}</target>                      <encoding>${project.build.sourceEncoding}</encoding>                      <compilerArgument>-Xlint:all</compilerArgument>                      <showWarnings>true</showWarnings>                      <showDeprecation>true</showDeprecation>                  </configuration>              </plugin>          </plugins>      </build>  </project> |

Напишем небольшой класс, который пишет что-нибудь в Log4j 1.2 (по этой библиотеку логирования у меня есть [отдельная статья](https://urvanov.ru/2019/07/04/%d0%bb%d0%be%d0%b3%d0%b8%d1%80%d0%be%d0%b2%d0%b0%d0%bd%d0%b8%d0%b5-%d1%81-log4j-1-2/)):

Slf4jLogbackLog4jExample.java

Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package ru.urvanov.javaexamples.slf4jlogbacklog4j;    import org.apache.log4j.Logger;    public class Slf4jLogbackLog4jExample {        private static final Logger logger = Logger.getLogger(              Slf4jLogbackLog4jExample.class);        public static void logToLog4j12() {          logger.debug("Log4j 1.2 example debug message");          logger.info("Log4j 1.2 example info message.");      }  } |

Соберём проект и положим его в локальный репозиторий Maven:

|  |  |
| --- | --- |
| 1 | mvn clean install |

Подключим собранный проект к нашему основному проекту:

pom.xml

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>ru.urvanov.javaexamples</groupId>      <artifactId>slf4jlogback-log4j-example</artifactId>      <version>0.0.1-SNAPSHOT</version>  </dependency> |

Построим дерево зависимостей и убедимся, что к нам в classpath заодно попадёт Log4j 1.2:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | $ mvn dependency:tree    [INFO] Scanning for projects...  [INFO]  [INFO] -----------< ru.urvanov.javaexamples:slf4j-logback-example >------------  [INFO] Building slf4j-logback-example 0.0.1-SNAPSHOT  [INFO] --------------------------------[ jar ]---------------------------------  [INFO]  [INFO] --- maven-dependency-plugin:2.8:tree (default-cli) @ slf4j-logback-example ---  [INFO] ru.urvanov.javaexamples:slf4j-logback-example:jar:0.0.1-SNAPSHOT  [INFO] +- junit:junit:jar:4.12:test  [INFO] |  \- org.hamcrest:hamcrest-core:jar:1.3:test  [INFO] +- ch.qos.logback:logback-classic:jar:1.2.3:compile  [INFO] |  \- ch.qos.logback:logback-core:jar:1.2.3:compile  [INFO] +- org.slf4j:slf4j-api:jar:1.7.26:compile  [INFO] +- ru.urvanov.javaexamples:slf4jlogback-commons-logging-example:jar:0.0.1-SNAPSHOT:compile  [INFO] +- org.slf4j:jcl-over-slf4j:jar:1.7.26:compile  [INFO] \- ru.urvanov.javaexamples:slf4jlogback-log4j-example:jar:0.0.1-SNAPSHOT:compile  [INFO]    \- log4j:log4j:jar:1.2.17:compile  [INFO] ------------------------------------------------------------------------  [INFO] BUILD SUCCESS  [INFO] ------------------------------------------------------------------------  [INFO] Total time:  1.549 s  [INFO] Finished at: 2019-07-08T12:02:31+03:00  [INFO] ------------------------------------------------------------------------ |

На 19 строке явно видно, что у нас в зависимостях подтянулся нежелательный Log4j. Его нужно убрать, а вместо него добавить log4j-over-slf4j:

pom.xml

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <dependency>      <groupId>ru.urvanov.javaexamples</groupId>      <artifactId>slf4jlogback-log4j-example</artifactId>      <version>0.0.1-SNAPSHOT</version>      <exclusions>          <exclusion>              <groupId>log4j</groupId>              <artifactId>log4j</artifactId>          </exclusion>      </exclusions>  </dependency>  <dependency>      <groupId>org.slf4j</groupId>      <artifactId>log4j-over-slf4j</artifactId>      <version>${slf4j.version}</version>  </dependency> |

В классе Slf4jLogbackExampleApp вызовем метод из добавленной библиотеки:

Java

|  |  |
| --- | --- |
| 1 | Slf4jLogbackLog4jExample.logToLog4j12(); |

Запустим класс Slf4jLogbackExampleApp и увидим строки из Slf4jLogbackLog4jExample:

|  |  |
| --- | --- |
| 1  2 | 08.07.2019 12:09:28.555 [main] DEBUG  r.u.j.s.Slf4jLogbackLog4jExample - Log4j 1.2 example debug message  08.07.2019 12:09:28.555 [main] INFO   r.u.j.s.Slf4jLogbackLog4jExample - Log4j 1.2 example info message. |

Конечный вариант файла “Slf4jLogbackExampleApp.java”:

Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package ru.urvanov.javaexamples.slf4jlogback;    import java.io.IOException;  import java.nio.file.Files;  import java.nio.file.Paths;    import org.slf4j.Logger;  import org.slf4j.LoggerFactory;    import ru.urvanov.javaexamples.slf4jlogbackcommonslogging.Slf4jLogbackCommonsLoggingExample;  import ru.urvanov.javaexamples.slf4jlogbacklog4j.Slf4jLogbackLog4jExample;    public class Slf4jLogbackExampleApp {      private static final Logger logger = LoggerFactory.getLogger(              Slf4jLogbackExampleApp.class);        private static final String FILENAME = "/file/does/not/exist";        public static void main(String[] args) {          logger.info("Just a log message.");          logger.debug("Message for debug level.");          try {              Files.readAllBytes(Paths.get(FILENAME));          } catch (IOException ioex) {              logger.error("Failed to read file {}.", FILENAME, ioex);          }          Slf4jLogbackCommonsLoggingExample.logCommonsLogging();          Slf4jLogbackLog4jExample.logToLog4j12();      }  } |

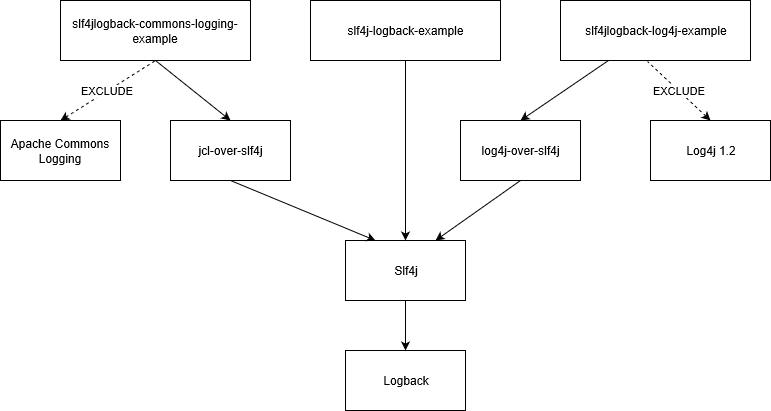
Конечный вариант файла “pom.xml” от slf4j-logback-example:

pom.xml

XHTML

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95 | <project xmlns="http://maven.apache.org/POM/4.0.0"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">      <modelVersion>4.0.0</modelVersion>        <groupId>ru.urvanov.javaexamples</groupId>      <artifactId>slf4j-logback-example</artifactId>      <version>0.0.1-SNAPSHOT</version>      <packaging>jar</packaging>        <name>slf4j-logback-example</name>      <url>http://maven.apache.org</url>        <properties>          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>          <java-version>1.8</java-version>          <slf4j.version>1.7.26</slf4j.version>      </properties>        <dependencies>          <dependency>              <groupId>junit</groupId>              <artifactId>junit</artifactId>              <version>4.12</version>              <scope>test</scope>          </dependency>            <!-- Logback -->          <dependency>              <groupId>ch.qos.logback</groupId>              <artifactId>logback-classic</artifactId>              <version>1.2.3</version>          </dependency>            <!-- Slf4j dependencies -->          <dependency>              <groupId>org.slf4j</groupId>              <artifactId>slf4j-api</artifactId>              <version>${slf4j.version}</version>          </dependency>          <dependency>              <groupId>org.slf4j</groupId>              <artifactId>jcl-over-slf4j</artifactId>              <version>${slf4j.version}</version>          </dependency>          <dependency>              <groupId>org.slf4j</groupId>              <artifactId>log4j-over-slf4j</artifactId>              <version>${slf4j.version}</version>          </dependency>            <!-- Other libraries -->          <dependency>              <groupId>ru.urvanov.javaexamples</groupId>              <artifactId>slf4jlogback-commons-logging-example</artifactId>              <version>0.0.1-SNAPSHOT</version>              <exclusions>                  <exclusion>                      <groupId>commons-logging</groupId>                      <artifactId>commons-logging</artifactId>                  </exclusion>              </exclusions>          </dependency>          <dependency>              <groupId>ru.urvanov.javaexamples</groupId>              <artifactId>slf4jlogback-log4j-example</artifactId>              <version>0.0.1-SNAPSHOT</version>              <exclusions>                  <exclusion>                      <groupId>log4j</groupId>                      <artifactId>log4j</artifactId>                  </exclusion>              </exclusions>          </dependency>        </dependencies>      <build>          <plugins>              <plugin>                  <groupId>org.apache.maven.plugins</groupId>                  <artifactId>maven-compiler-plugin</artifactId>                  <version>2.5.1</version>                  <configuration>                      <source>${java-version}</source>                      <target>${java-version}</target>                      <encoding>${project.build.sourceEncoding}</encoding>                      <compilerArgument>-Xlint:all</compilerArgument>                      <showWarnings>true</showWarnings>                      <showDeprecation>true</showDeprecation>                  </configuration>              </plugin>          </plugins>      </build>    </project> |

В результате логирование в нашем проекте можно отразить следующей схемой:

Slf4j Logback example

Вот такая ситуация с логированием/журналированием в Java на текущий момент.