Forcing Tomcat to log through SLF4J/Logback

So you have your executable web application in JAR with bundled Tomcat [http://nurkiewicz.blogspot.com/2012/11/standalone-web-application-with.html] (make sure to read that one first). However there are these annoying Tomcat logs at the beginning, independent from our application logs and not customizable:

```
Nov 24, 2012 11:44:02 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Nov 24, 2012 11:44:02 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Tomcat
Nov 24, 2012 11:44:02 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.30
Nov 24, 2012 11:44:05 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
```

I would really like to quite them down, or even better save them somewhere since they sometimes reveal important failures. But I definitely don't want to have a separate `java.util.logging` configuration. Did you wonder after reading the previous article [http://nurkiewicz.blogspot.com/2012/11/standalone-web-application-with.html] how did I knew that runnable Tomcat JAR supports `-httpPort` parameter and few others? Well, I checked the sources, but it's simpler to just ask for help:

```
$ java -jar target/standalone.jar -help
usage: java -jar [path to your exec war jar]
 -ajpPort <ajpPort>                     ajp port to use
 -clientAuth                            enable client authentication for
                                        https
 -D <arg>                               key=value
 -extractDirectory <extractDirectory>   path to extract war content,
                                        default value: .extract
 -h,--help                              help
 -httpPort <httpPort>                   http port to use
 -httpProtocol <httpProtocol>           http protocol to use: HTTP/1.1 or
                                        org.apache.coyote.http11.Http11Nio
                                        Protocol
 -httpsPort <httpsPort>                 https port to use
 -keyAlias <keyAlias>                   alias from keystore for ssl
 -loggerName <loggerName>               logger to use: slf4j to use slf4j
                                        bridge on top of jul
 -obfuscate <password>                  obfuscate the password and exit
 -resetExtract                          clean previous extract directory
 -serverXmlPath <serverXmlPath>         server.xml to use, optional
 -uriEncoding <uriEncoding>             connector uriEncoding default
                                        ISO-8859-1
 -X,--debug                             debug
```

The `-loggerName` parameter looks quite promising. First try:
[]

```
$ java -jar target/standalone.jar -loggerName slf4j
WARNING: issue configuring slf4j jul bridge, skip it
```

No good. Quick look at the source code again and it turns out that SLF4J library is missing. Since this parameter is interpreted during Tomcat bootstrapping (way before web application is deployed), `slf4j-api.jar` inside my web application is not enough, it has to be available for root class loader (equivalent to `/lib` directory in packaged Tomcat). Luckily plugin exposes `<extraDependencies/>` configuration parameter [http://tomcat.apache.org/maven-plugin-2.0/tomcat7-

maven-plugin/exec-war-only-mojo.html#extraDependencies] :

```
<configuration>
    <path>/standalone</path>
    <enableNaming>false</enableNaming>
    <finalName>standalone.jar</finalName>
    <charset>utf-8</charset>
    <extraDependencies>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.2</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jul-to-slf4j</artifactId>
            <version>1.7.2</version>
        </dependency>
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>1.0.7</version>
        </dependency>
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-core</artifactId>
            <version>1.0.7</version>
        </dependency>
    </extraDependencies>
</configuration>
```

Running Tomcat and... success!

```
00:01:27.110 [main] INFO  o.a.coyote.http11.Http11Protocol - Initializing ProtocolHandler ["http-bio-8080"]
00:01:27.127 [main] INFO  o.a.catalina.core.StandardService - Starting service Tomcat
00:01:27.128 [main] INFO  o.a.catalina.core.StandardEngine - Starting Servlet Engine: Apache Tomcat/7.0.33
00:01:29.645 [main] INFO  o.a.coyote.http11.Http11Protocol - Starting ProtocolHandler ["http-bio-8080"]
```

Well, not quite. If you use Logback [http://logback.qos.ch/] on a daily basis you are familiar with default console logging pattern. We are not picking up any `logback.xml`. From my experience it seems that placing `logback.xml` externally somewhere in your file system is superior to putting it inside your binary, especially with auto refreshing feature turned on:

```
<configuration scan="true" scanPeriod="5 seconds">
    <!-- ... -->
</configuration>
```

Put some fallback `logback.xml` file in the root of your CLASSPATH in case no other file was specified like below and voilà:

```
$ java -jar standalone.jar -httpPort=8081 -loggerName=slf4j \
    -Dlogback.configurationFile=/etc/foo/logback.xml
```

Finally, clean and consistent logging, most likely to a single file.