Logging

Contents

Logging

By default, logging functionality is provided by the clojure.tools.logging library. The library provides macros that delegate to a specific logging implementation. The default implementation used in Luminus is the logback library.

There are six log levels in `clojure.tools.logging`, and any Clojure data structures can be logged directly. The log levels are `trace`, `debug`, `info`, `warn`, `error`, and `fatal`.

```
(ns example
 (:require [clojure.tools.logging :as log]))

(log/info "Hello")
=>[2015-12-24 09:04:25,711][INFO][myapp.handler] Hello

(log/debug {:user {:id "Anonymous"}})
=>[2015-12-24 09:04:25,711][DEBUG][myapp.handler] {:user {:id "Anonymo

(log/error (Exception. "I'm an error") "something bad happened")
=>[2015-12-24 09:43:47,193][ERROR][myapp.handler] something bad happen
  java.lang.Exception: I'm an error
        at myapp.handler$init.invoke(handler.clj:21)
        at myapp.core$start_http_server.invoke(core.clj:44)
        at myapp.core$start_app.invoke(core.clj:61)
        ...
```

Logging Configuration

The default logger configuration is found in the `resources/logback.xml` file and looks as follows:

Build Tool: lein ▾

Topics

Books

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <statusListener class="ch.qos.logback.core.status.NopStatusListene
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender
        <encoder>
            <charset>UTF-8</charset>
            <pattern>%date{ISO8601} [%thread] %-5level %logger{36} - %
        </encoder>
    </appender>
    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFi
        <file>log/myapp.log</file>
        <rollingPolicy
         class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>log/myapp.%d{yyyy-MM-dd}.%i.log</fileName
            <timeBasedFileNamingAndTriggeringPolicy
             class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP"
                <maxFileSize>100MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <!-- keep 30 days of history -->
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <charset>UTF-8</charset>
            <pattern>%date{ISO8601} [%thread] %-5level %logger{36} - %
        </encoder>
    </appender>
    <root level="INFO">
        <appender-ref ref="STDOUT" />
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

An external logging configuration can be provided by setting the `logback.configurationFile` Java system property that points to the path of the log configuration file. For example, we could create a production configuration called `prod-log-config.xml` and have it log to the `/var/log/myapp.log` location.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <statusListener class="ch.qos.logback.core.status.NopStatusListene
    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFi
        <file>/var/log/myapp.log</file>
        <rollingPolicy
         class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>/var/log/myapp.%d{yyyy-MM-dd}.%i.log</fil
            <timeBasedFileNamingAndTriggeringPolicy
             class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP"
                <maxFileSize>100MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <!-- keep 30 days of history -->
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder>
            <charset>UTF-8</charset>
            <pattern>%date{ISO8601} [%thread] %-5level %logger{36} - %
        </encoder>
    </appender>
    <root level="INFO">
        <appender-ref ref="FILE" />
```

```
    </root>
</configuration>
```

Then we can start the app with the following flag to have it use this logging configuration:

```
java -Dlogback.configurationFile=prod-log-config.xml -jar myapp.jar
```

Please refer to the official documentation for further information on configuring `logback`.
