DoTTed, LLC.

“When you’re in doubt, we figure it out”

SysMon+

PC Performance Monitoring for Windows

Build Systems Document

MSCS710 Project

Spring 2024

Written By: Benjamin Fiore, Alec Lehmphul, and Nicholas Petrilli

Version History

Version	Authors	Description	Date
1.0.0	Nick Petrilli, Alec Lehmphul, Ben Fiore	Build System Document - Initial Version	3/22/2024

Table of Contents

Objective 4
Building React.js Frontend with NPM 5
Building Java Spring Boot Backend with Maven 7
Packaging React.js Frontend 10
Packaging Java Spring Boot Backend with Maven 12
References 15

Objective

SysMon+ provides a secure, single-device metrics collection and visualization by storing all collected metrics in a database that is installed and configured on the end user's device (the installation and configuration of the database is handled by the installation of the application, requiring no manual database setup by the end user). When the user is ready to view the latest collected metrics, they can open the dashboard to see this data, as well as any data updates received by *SysMon+* in real time.

Building React.js Frontend with NPM

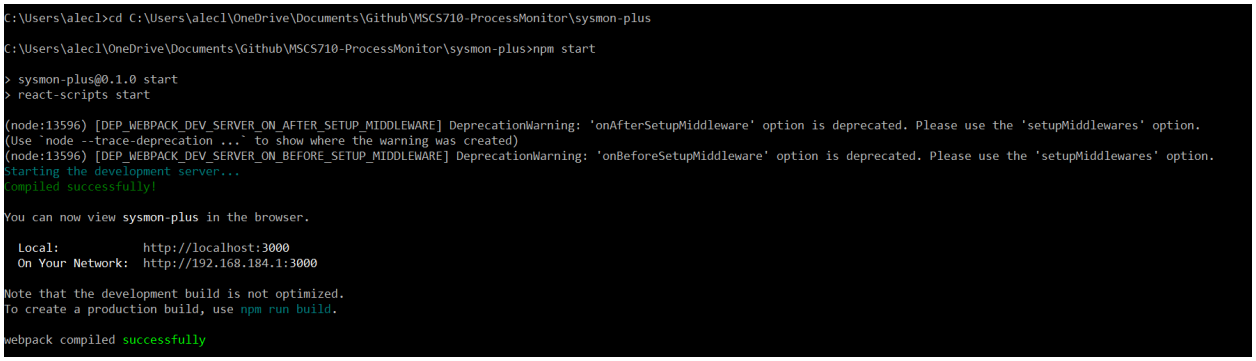
In order to package and build the React frontend, a package.json file must be present. This file contains descriptive and functional metadata about a project, such as a name, version, and dependencies. It essentially provides the npm package manager with vital information to help identify the project and handle the project's dependencies.

Below I list some important tags contained within SysMon+'s package.json file.

- The "name" tag specifies the name of the application.
- The "dependencies" tag contains all the necessary dependencies that SysMon+ relies on. This includes all the native React libraries and the chart.js libraries.
    - Note that all dependencies were installed using the command 'npm install {dependency name}'
- The "scripts" tag is a dictionary containing script commands that are run at various times in the life cycle of SysMon+'s package. The key is the lifecycle event like start or build, and the value is the command to run at that point.

The `npm start` command is how we build the React.js frontend. This command invokes the `react-scripts start` command, which starts the development server, opens the application in the default web browser, and watches for any file changes within the project. As a result, it provides a live-reloading feature that automatically refreshes the page whenever changes to the code are saved, facilitating a smoother development experience.

Below is a screenshot of a successful build message.



After this command, the React.js frontend is accessible on localhost on port 3000 → 'http://localhost:3000' as a dynamic website.

Building Java Spring Boot Backend with Maven

Below are screenshots of our pom.xml file. This pom.xml file is the project's configuration file that defines the project's build settings with necessary dependencies and plugins. The <project> element is the root element of the POM which specifies the XML namespace for our Maven model version, 4.0.0. The <parent> element defines the parent project in which the current project inherits configurations and plugins. The bottom <properties> defines which versions of Java and JNA the project uses, which is 17 and 5.14.0 respectively.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.process_monitor</groupId>
    <artifactId>process-monitor</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>process-monitor</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>17</java.version>
        <jna.version> 5.14.0</jna.version>
    </properties>
```

Next in the pom.xml file are our required dependencies for the project. The first one listed is a dependency on the SQLite JDBC driver library, which allows our project to connect to our SQLite database. The dialects in Hibernate are responsible for translating our queries into database-specific SQL dialects, in our case SQLite. The spring boot starter web provides dependencies for creating a web application, using the Spring MVC framework and the embedded Apache Tomcat web server. The last dependency in the screenshot is oshi-core, (Operating System Hardware Information) which is the library that allows us to collect our system metrics.

```
<dependencies> Add Spring Boot Starters...
    <dependency>
        <groupId>org.xerial</groupId>
        <artifactId>sqlite-jdbc</artifactId>
        <version>3.45.1.0</version>
    </dependency>
    <dependency>
        <groupId>org.hibernate.orm</groupId>
        <artifactId>hibernate-community-dialects</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>com.github.oshi</groupId>
        <artifactId>oshi-core</artifactId>
        <version>6.4.11</version>
    </dependency>
```

The spring boot dev tools configures the project to have automatic restarts when changes are made, along with live reload when new changes are saved to our files. The spring boot starter test provides us with essential testing libraries such as JUnit that we will be using to test our application. Lastly, the spring boot maven plugin allows us to run our application with the spring-boot: run command during development. For packaging, this plugin packages the Spring Boot application into an executable JAR file, containing all the necessary dependencies and resources required to run the application.

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

    <dependency>
      <groupId>com.github.oshi</groupId>
      <artifactId>oshi-demo</artifactId>
      <version>4.9.5</version>
    </dependency>
</dependencies>

     <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

Packaging React.js Frontend

In order to package *SysMon+*'s frontend environment and prepare it for building into its production version, there are a series of processes that must be completed by the application's package manager, NPM.

Task 1: Build the application

Within the development environment terminal, the command "npm run build" is executed to produce a production-ready build of *SysMon+* from its development code. This command is executed within the frontend directory of the development environment, as the frontend environment has access to the "react-scripts" package that allows for the necessary commands to develop a React.js application.

```
PS [redacted] npm run build

> sysmon-plus@0.1.0 build
> react-scripts build

Creating an optimized production build...
```

Following the execution of the build command, the "react-scripts" package reports on the command's success, and informs the developer that the build is ready to be deployed to a static server. The configuration of said static server will be discussed later on in this document.

```
The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:
```

Task 2: Install The Static Server

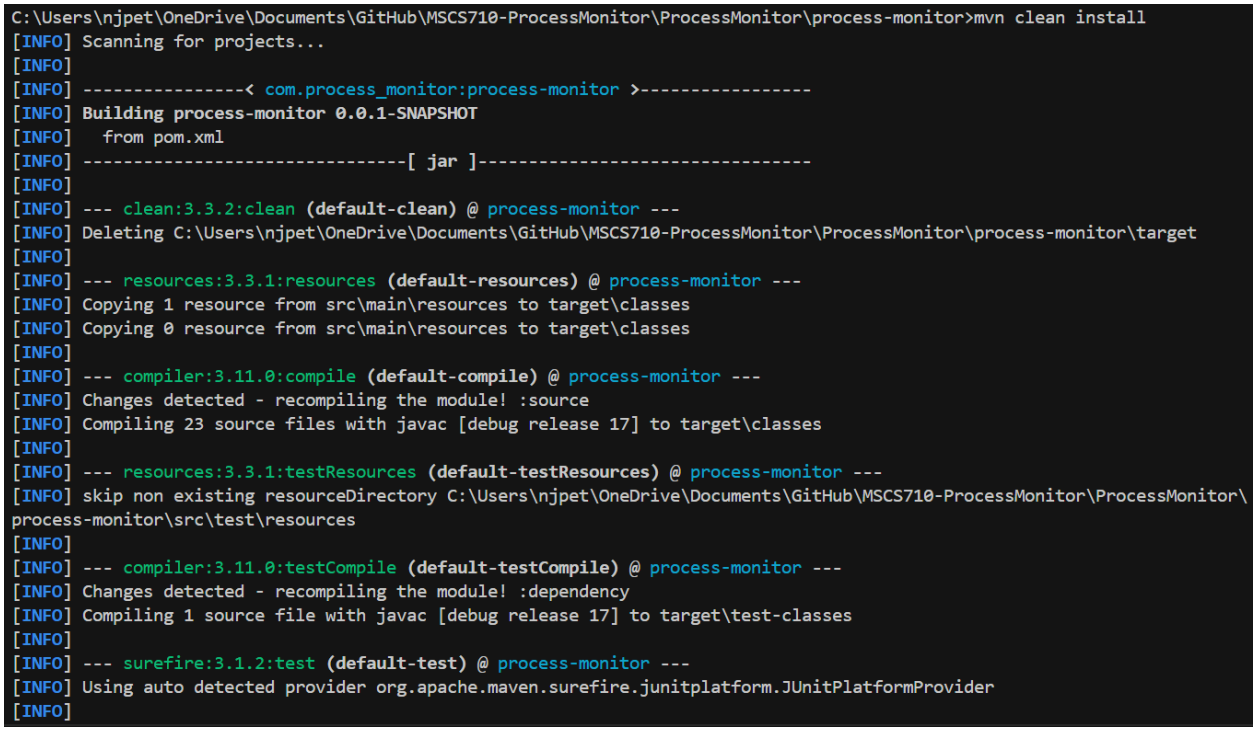
In order to utilize a static server, its package must be installed on the user's device. This package, named "serve", must be installed in order to utilize the production build of *SysMon+* on end users' devices. To install this package, the command "npm install -g serve" must be executed within the user device's terminal.

NOTE: Running this command will require that the end user's device has Node.js/NPM installed as well.

Provided no errors are reported in the terminal, running this command will result in the installation of the npm “serve” package.

Packaging Java Spring Boot Backend with Maven

To package our Java Spring Boot backend, we run the `mvn clean install` command to generate the executable JAR file. This command performs multiple Maven commands in sequence. First the project directory cleans any previous compiled versions and ensures there is a clean slate for this build. Next, all files are compiled and tested and can only succeed if none of the tests fail. Then, the compiled code and other resources are packaged into a JAR file within the target directory. Lastly, the install means the JAR file is copied to the local Maven repository, allowing other projects to use it as a dependency directly. The following images show the command running and succeeding:



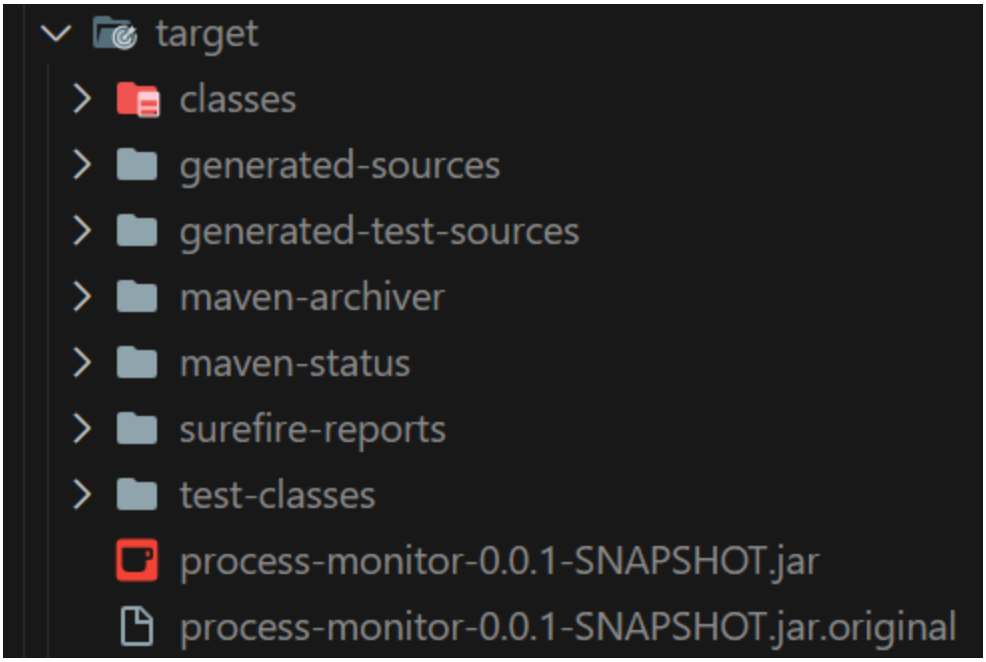
```
SLF4J(W): Found provider [org.slf4j.simple.SimpleServiceProvider@662ac478]
SLF4J(W): See https://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J(I): Actual provider is of type [ch.qos.logback.classic.spi.LogbackServiceProvider@31190526]
18:50:05.754 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect d
efault configuration classes for test class [com.process_monitor.processmonitor.ProcessMonitorApplicationTests]: ProcessMo
nitorApplicationTests does not declare any static, non-private, non-final, nested classes annotated with @Configuration.
18:50:05.894 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper -- Found @SpringBootConfi
guration com.process_monitor.processmonitor.ProcessMonitorApplication for test class com.process_monitor.processmonitor.Pr
ocessMonitorApplicationTests
18:50:06.173 [main] INFO org.springframework.boot.devtools.restart.RestartApplicationListener -- Restart disabled due to c
ontext in which it is running

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.2.2)

2024-03-23T18:50:06.644-04:00  INFO 6272 --- [           main] c.p.p.ProcessMonitorApplicationTests     : Starting Process
MonitorApplicationTests using Java 17.0.4.1 with PID 6272 (started by njpet in C:\Users\njpet\OneDrive\Documents\GitHub\MS
CS710-ProcessMonitor\ProcessMonitor\process-monitor)
2024-03-23T18:50:06.646-04:00  INFO 6272 --- [           main] c.p.p.ProcessMonitorApplicationTests     : No active profil
e set, falling back to 1 default profile: "default"
2024-03-23T18:50:09.383-04:00  INFO 6272 --- [           main] c.p.p.ProcessMonitorApplicationTests     : Started ProcessM
onitorApplicationTests in 3.22 seconds (process running for 4.702)
2024-03-23T18:50:09.403-04:00  INFO 6272 --- [taskScheduler-1] c.p.p.collector.MetricCollector         : Metric Collectio
n performed at 2024-03-23T18:50:09.403293400
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been a
ppended
```

```
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ process-monitor ---
[INFO] Building jar: C:\Users\njpet\OneDrive\Documents\GitHub\MSCS710-ProcessMonitor\ProcessMonitor\process-monitor\target
\process-monitor-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.2.2:repackage (repackage) @ process-monitor ---
[INFO] Replacing main artifact C:\Users\njpet\OneDrive\Documents\GitHub\MSCS710-ProcessMonitor\ProcessMonitor\process-moni
tor\target\process-monitor-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to C:\Users\njpet\OneDrive\Documents\GitHub\MSCS710-ProcessMonitor\ProcessMo
nitor\process-monitor\target\process-monitor-0.0.1-SNAPSHOT.jar.original
[INFO]
[INFO] --- install:3.1.1:install (default-install) @ process-monitor ---
[INFO] Installing C:\Users\njpet\OneDrive\Documents\GitHub\MSCS710-ProcessMonitor\ProcessMonitor\process-monitor\pom.xml t
o C:\Users\njpet\.m2\repository\com\process_monitor\process-monitor\0.0.1-SNAPSHOT\process-monitor-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\Users\njpet\OneDrive\Documents\GitHub\MSCS710-ProcessMonitor\ProcessMonitor\process-monitor\target\pr
ocess-monitor-0.0.1-SNAPSHOT.jar to C:\Users\njpet\.m2\repository\com\process_monitor\process-monitor\0.0.1-SNAPSHOT\proce
ss-monitor-0.0.1-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  29.348 s
[INFO] Finished at: 2024-03-23T18:50:29-04:00
[INFO] ------------------------------------------------------------------------
```

Following the build success, we can browse to the target folder and see our executable JAR called process-monitor-0.0.1-SNAPSHOT.jar:


target
classes
generated-sources
generated-test-sources
maven-archiver
maven-status
surefire-reports
test-classes
process-monitor-0.0.1-SNAPSHOT.jar
process-monitor-0.0.1-SNAPSHOT.jar.original

References

For more on the NPM, refer to the link below.
https://www.npmjs.com/

For Spring Boot documentation and implementation, refer to the link below.
https://spring.io/projects/spring-boot

For Apache Maven documentation, refer to the link below.
https://maven.apache.org/

For more on Apache Maven Repository, refer to the link below.
https://mvnrepository.com/repos