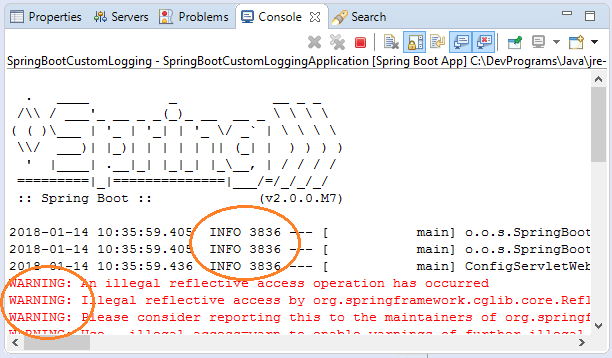
**Лабораторна робота № 22 «Використання в Spring Boot логерів. Застосування Spring Boot Actuator»**

**"Logging"** це  **"запис"**  всіх проблем під час роботи програми. Проблемами тут можуть бути інформація ошбки, попередження (warning), та інша інформація. Ця інформація може бути відображена на екрані **Console** або записана у файлі.

Коли ви запускаєте програму  **Spring Boot**  безпосередньо на **Eclipse**  ви можете побачити інформацію на вікні  **Console**, дана інформація дає вам знати статус програми, помилки, що відбуваються під час роботи програми. Це і є  **Logging**!



Є багато різних бібліотек  **Logging** як  **Logback**, **Log4j2**, **JUL**. **Spring Boot**  може працювати з усіма цими бібліотеками. За промовчанням  **Spring Boot**  вже автоматично конфігурований і використовує бібліотеку  **Logback** для його  **logging** , і вам не потрібно додаткових конфігурацій, тільки якщо ви хочете кастомізувати деякі аргументи (argument). Кастомізації тут можуть визначити назву файлу для запису інформації, шаблон (pattern).

У цій статті ми обговоримо принцип роботи  **Logback** у додатку  **Spring Boot** і вивчимо спосіб кастомізації для  **Logback** .

Властивості (properties), які можна кастомізувати:

logging.config

logging.exception-conversion-word

logging.file

logging.level.\*

logging.path

logging.pattern.console

logging.pattern.file

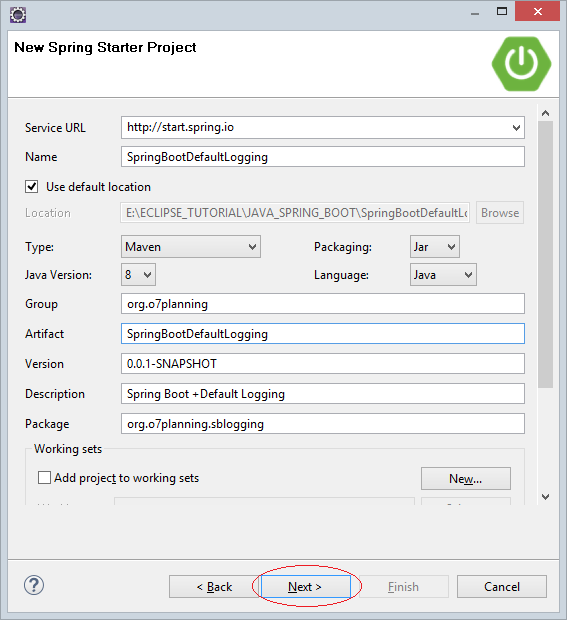
logging.pattern.level

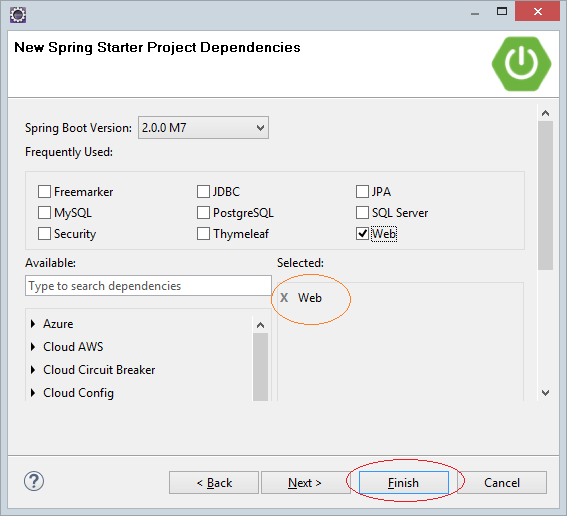
logging.register-shutdown-hook

## 

## 2- Створити проект Spring Boot

На  **Eclipse** створіть проект  **Spring Boot**.





Щоб використовувати  **Logging** , вам потрібно використовувати один з наступних **"Starter"** :

* **spring-boot-starter-logging**
* **spring-boot-starter-web**

pom.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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.o7planning</groupId>

<artifactId>SpringBootDefaultLogging</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringBootDefaultLogging</name>

<description>Spring Boot +Default Logging</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.0.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

MainController.java

package org.o7planning.sblogging.controller;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseBody;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Controller

public class MainController {

private static final Logger LOGGER = LoggerFactory.getLogger(MainController.class);

@ResponseBody

@RequestMapping(path = "/")

public String home() {

LOGGER.trace("This is TRACE");

LOGGER.debug("This is DEBUG");

LOGGER.info("This is INFO");

LOGGER.warn("This is WARN");

LOGGER.error("This is ERROR");

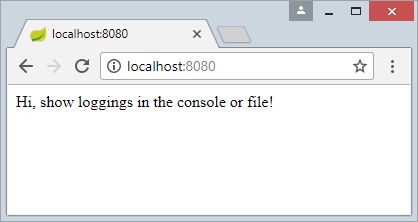
return "Hi, show loggings in the console or file!";

}

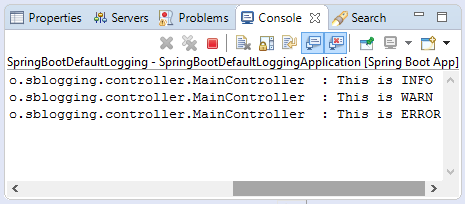
}

Запустіть ваш додаток безпосередньо на **Spring Boot** , потім пройдіть за наступним посиланням:

* <http://localhost:8080/>



На вікні  **Console** у **Eclipse**  ви можете побачити інформацію **Logs** таким чином:



2018-01-14 13:21:57.593 INFO 7980 --- [nio-8080-exec-6] o.o.sblogging.controller.MainController : This is INFO

2018-01-14 13:21:57.593 WARN 7980 --- [nio-8080-exec-6] o.o.sblogging.controller.MainController : This is WARN

2018-01-14 13:21:57.593 ERROR 7980 --- [nio-8080-exec-6] o.o.sblogging.controller.MainController : This is ERROR

## 

## 3- Logging Level

Основуючи рівень серйозності проблеми, **Logback**  ділить інформацію для запису на 5 рівнів (Level), найменш серйозна це  **TRACE** , і найсерйозніша це  **ERROR** . Примітка: Є деякі  **Logging бібліотеки,**  які ділять інформацію для запису на 7 різних рівнів.

1. TRACE
2. DEBUG
3. INFO
4. WARN
5. ERROR
6. FATAL
7. OFF

За замовчуванням  **Spring Boot**  запсує лише інформацію із серйозністю від рівня  **INFO** та вище

# Default:

logging.level.root=INFO

Змінити  **Logging Level** у **application.properties** :

\* application.properties \*

logging.level.root=WARN

# ..

Перезапустіть вашу програму, і перегляньте інформацію записану на вікні  **Console** :

2018-01-14 17:45:50.341 WARN 8500 --- [nio-8080-exec-1] o.o.sblogging.controller.MainController : This is WARN

2018-01-14 17:45:50.342 ERROR 8500 --- [nio-8080-exec-1] o.o.sblogging.controller.MainController : This is ERROR

## 

## 4- Logging File

 За замовчуванням  інформація  **про loging записана на екрані Console** , але ви можете конфігурувати, щоб вони були записані у файлах.

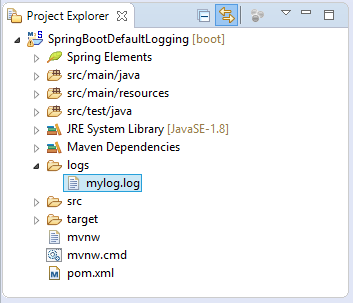
\* application.properties \*

logging.file=logs/mylog.log

Перезапустіть вашу програму безпосередньо на  **Eclipse** , потім пройдіть за наступним посиланням:

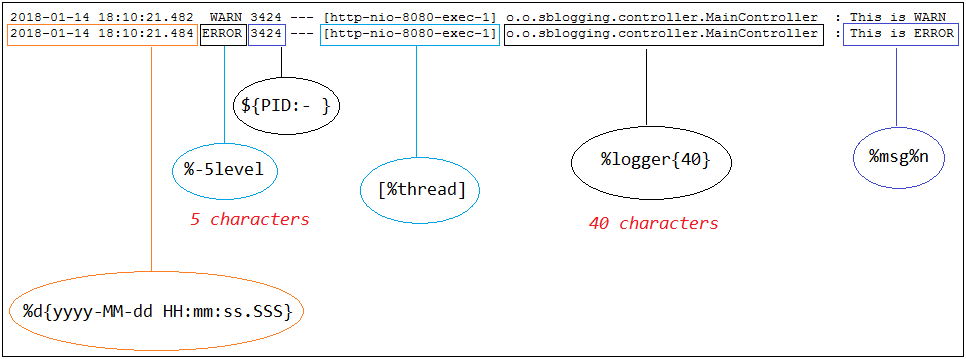
* <http://localhost:8080/>

Потім оновіть проект, ви можете побачити створений файл  **logging** .



## 5- Logging Pattern

Записи Log (Log record) записані за шаблоном (pattern), нижче є шаблон (pattern) за замовчуванням:



І ви можете змінити  **"Logging pattern"**  кастомізуючи властивості (properties) нижче:

1. **logging.pattern.console**
2. **logging.pattern.file**

# Pattern:

logging.pattern.console= %d{yyyy-MMM-dd HH:mm:ss.SSS} %-5level [%thread] %logger{15} - %msg%n

# Output:

2018-Jan-17 01:58:49.958 WARN [http-nio-8080-exec-1] o.o.s.c.MainController - This is WARN

2018-Jan-17 01:58:49.960 ERROR [http-nio-8080-exec-1] o.o.s.c.MainController - This is ERROR

# Pattern:

logging.pattern.console= %d{dd/MM/yyyy HH:mm:ss.SSS} %-5level [%thread] %logger{115} - %msg%n

# Output:

17/01/2018 02:15:15.052 WARN [http-nio-8080-exec-1] org.o7planning.sblogging.controller.MainController - This is WARN

17/01/2018 02:15:15.054 ERROR [http-nio-8080-exec-1] org.o7planning.sblogging.controller.MainController - This is ERROR

# Pattern:

logging.pattern.console=%d{yy-MMMM-dd HH:mm:ss:SSS} %5p %t %c{2}:%L - %m%n

# Output:

18-January-17 02:21:20:317 WARN http-nio-8080-exec-1 o.o.s.c.MainController:22 - This is WARN

18-January-17 02:21:20:320 ERROR http-nio-8080-exec-1 o.o.s.c.MainController:23 - This is ERROR