

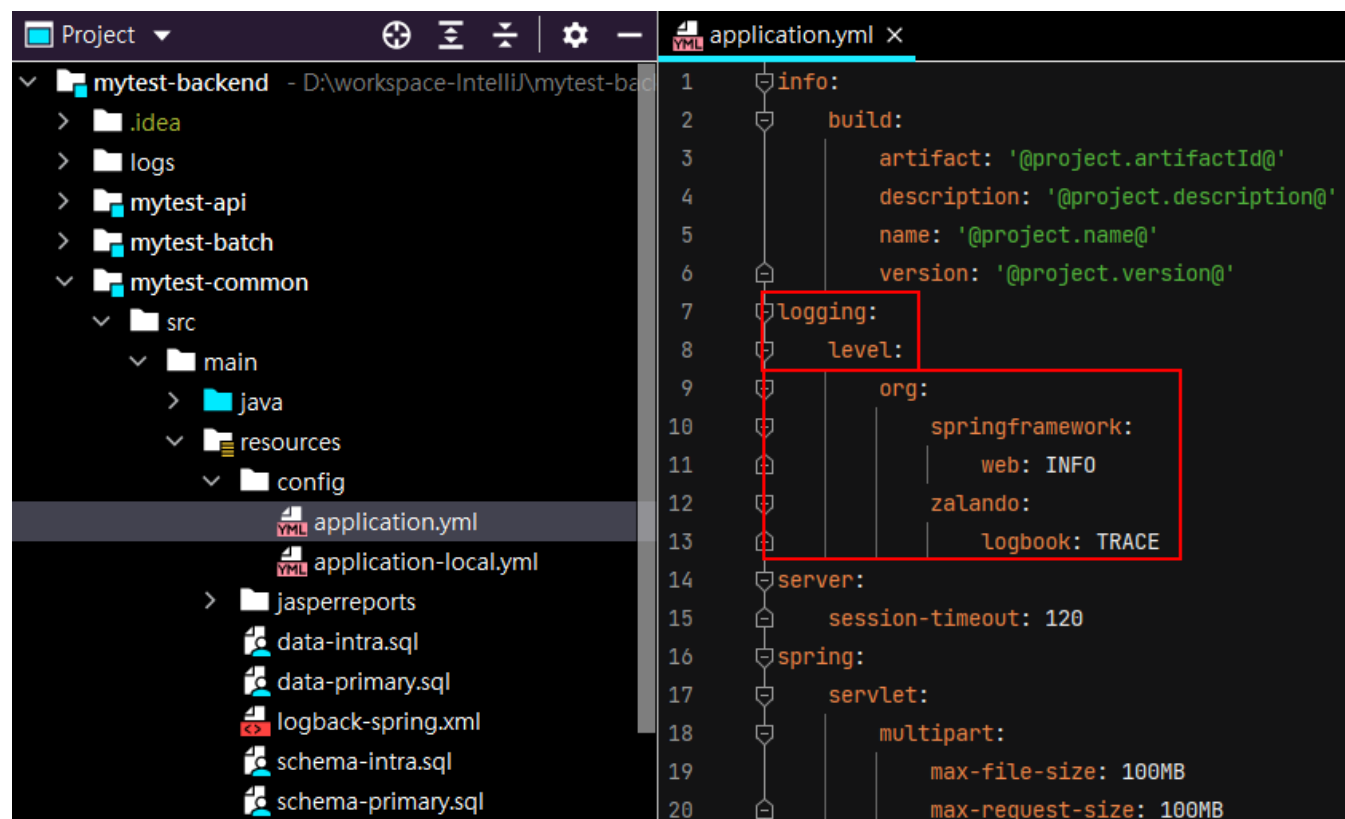
使用 Logback

application.yml 配置

Spring Boot 預設使用的日誌框架是 Logback，所以可以透過 application.yml 進行相關配置

在 [Spring Boot - Common Application properties](#) 內的 Core properties 的章節，可以找到 Logging 相關的通用設定

目前，我們是只使用「logging.level.*」，其餘更多的設定，建議是直接看 Logback 的設定檔（logback-spring.xml）



```
1 info:
2   build:
3     artifact: '@project.artifactId@'
4     description: '@project.description@'
5     name: '@project.name@'
6     version: '@project.version@'
7   logging:
8     level:
9   org:
10    springframework:
11      web: INFO
12    zalando:
13      logbook: TRACE
14  server:
15    session-timeout: 120
16  spring:
17    servlet:
18      multipart:
19        max-file-size: 100MB
20        max-request-size: 100MB
```

logback-spring.xml 配置

檔案位置 `D:\workspace-IntelliJ\mytest-backend\mytest-common\src\main\resources\logback-spring.xml`

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- <configuration> scan true -->
<configuration scan="true">
    <statusListener class="ch.qos.logback.core.status.NopStatusListener"/>

    <!-- console -->
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>[%d{YYYY-MM-dd HH:mm:ss.SSS}][%10.10thread][%highlight(%-5level)][%cyan(%-30.30logger{36})]
[%4L] - %msg%n
            </pattern>
        </encoder>
    </appender>

    <!-- file -->
    <appender name="RollingFile" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>logs/log.log</file>
        <encoder>
            <charset>UTF-8</charset>
            <pattern>[%d{YYYY-MM-dd HH:mm:ss.SSS}][%10.10thread][%highlight(%-5level)][%-30.30logger{36}][%4L] - %msg%n
            </pattern>
        </encoder>

        <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
            <fileNamePattern>logs/log.%i.log</fileNamePattern>
            <minIndex>1</minIndex>
            <maxIndex>10</maxIndex>
        </rollingPolicy>
        <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
            <maxFileSize>10MB</maxFileSize>
        </triggeringPolicy>
    </appender>

    <!-- ROOT Log INFOWARNERROR -->
    <root level="info">
        <appender-ref ref="STDOUT"/>
        <appender-ref ref="RollingFile"/>
    </root>

    <springProfile name="default">
        <root level="info">
            <appender-ref ref="STDOUT"/>
            <appender-ref ref="RollingFile"/>
        </root>
    </springProfile>

    <springProfile name="local">
        <root level="info">
            <appender-ref ref="STDOUT"/>
            <appender-ref ref="RollingFile"/>
        </root>
    </springProfile>

</configuration>

```



Log 內容等級由低至高：TRACE、DEBUG、INFO、WARN、ERROR

如何優雅地將資訊寫入 Log

使用 Lombok 的註解：@Slf4j

自動生成該類的 log 靜態常量，要打日誌就可以直接打，不用再手動 new log 靜態量了

```
@Slf4j
public class User {
    public static void main(String[] args) {
        log.info("hello");
    }
}
```

=

```
public class User {

    private static final Logger log = LoggerFactory.getLogger(User.class);

    public static void main(String[] args) {
        log.info("hello");
    }
}
```

除了 @Slf4j 之外，lombok 也提供其他日誌框架的變種注解可以用，像是 @Log、@Log4j...等，他們都是幫我們創建一個靜態常量 log，只是使用的 library 不一樣

如何將try-catch捕捉到的異常寫入 Log

```
try {
    // TODO more code here
} catch (IOException ioe) {
    //
    log.error("", ioe);
    log.error("Error reading configuration file", ioe);
    log.error(toString() + "_" + ioe.getMessage(), ioe);
}
```

不合適的寫法

```
try {
    Integer x = null;
    ++x;
} catch (Exception e) {
    //
    log.error("" + e);
    log.error(e.toString());
    log.error(e.getMessage());
    log.error(null, e);
    log.error("{} ", e.getMessage());
    log.error("Error reading configuration file: " + e);
    log.error("Error reading configuration file: " + e.getMessage());
}
```