JDBC 관련 접근 log로 남기기!

Spring Boot를 실행하면 로그가 나오는 것을 볼 수 있습니다.
Console View에 말입니다.
Spring Boot는 기본적으로 Logback을 포함하기 때문입니다.
하지만! Console에서의 Log는 의미가 없습니다.

서버에 문제가 발생한 경우 이를 확인할 방법이 없습니다.
이를 특정 폴더에 파일 형태로 남기게 되고 이를 날짜별로 분류하거나 용량별로 구분합니다.

1. DB설정 정보 수정하기 및 application.properties 수정하기

```
# log4jdbc를 추가한 형태의 DB 정보이다.
# DB에 접근한 정보를 log로 남길 수 있다.
# 또한 기존 application.properties 에서 별도로 작성하여 Bean객체화 했다.
# 이는 민감정보인 DB접근정보를 따로 관리하기 위함이다.
# 현재는 classpath에 올려놨지만 따로 빼서 관리하도록 한다.

# spring.datasource.url=jdbc:mysql://localhost:3307/testdb?serverTimezone=UTC&characterEncoding=UTF-8
# spring.datasource.username=root
# spring.datasource.password=1234
# spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver


spring.datasource.url=jdbc:log4jdbc:mysql://localhost:3307/testdb?serverTimezone=UTC&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=1234
spring.datasource.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy
```

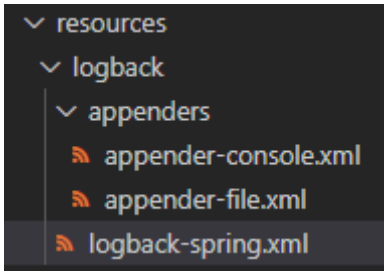
기존에 작성했던 Config.properties를 수정합니다.

```
spring.mvc.view.prefix: /WEB-INF/views/
spring.mvc.view.suffix: .jsp

logging:
    path: c:/dev/log/spring
    file: log-file

---

spring:
    profiles: log-windows

logging:
    path: c:/dev/log/spring

---

spring:
    profiles: log-linux

logging:
    path: /log/spring/
```

또한 application.properties를 application.yml파일로 바꾸어줍니다!
각각의 프로파일에 대한 logging path를 정의해준다고 합니다.

2. logback 관련 설정파일 작성!

설정파일 경로



logback-spring.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration scan="true" scanPeriod="10 seconds">

    <conversionRule conversionWord="clr" converterClass="org.springframework.boot.logging.logback.ColorConverter" />

    <springProfile name="log-jdbc">
        <logger name="jdbc" level="OFF"/>
        <logger name="jdbc.sqlonly" level="OFF"/>
        <logger name="jdbc.sqltiming" level="DEBUG"/>
        <logger name="jdbc.audit" level="OFF"/>
        <logger name="jdbc.resultset" level="OFF"/>
        <logger name="jdbc.resultsettable" level="DEBUG"/>
        <logger name="jdbc.connection" level="OFF"/>
    </springProfile>

    <springProfile name="log-file">
        <include resource="appenders/appender-file.xml" />
          <root level="INFO">
            <appender-ref ref="FILE" />
          </root>
    </springProfile>

    <springProfile name="log-console">
        <include resource="appenders/appender-console.xml" />
          <root level="INFO">
            <appender-ref ref="CONSOLE" />
          </root>
    </springProfile>
</configuration>
```

appender-console.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<included>
    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{MM-dd HH:mm:ss.SSS} %clr(${LOG_LEVEL_PATTERN:-%5p}) [%t] %clr(%-40.40logger{39}){cyan} | %msg%n</pattern>
        </encoder>
    </appender>
</included>
```

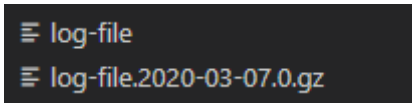
appender-file.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<included>
    <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_PATH}/${LOG_FILE}.log</file>
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
            <fileNamePattern>${LOG_PATH}/${LOG_FILE}.%d{yyyy-MM-dd}.%i.log</fileNamePattern>
            <maxFileSize>${LOG_FILE_MAX_SIZE:-10MB}</maxFileSize>
            <maxHistory>${LOG_FILE_MAX_HISTORY:-0}</maxHistory>
        </rollingPolicy>
    </appender>
</included>
```

로그를 남기는 형태를 정의해줍니다.
이와같이 설정해줍니다. 자세한 이해를 하지는 못해서 상세내용은 추후에 추가하겠습니다.

3. Log확인!!

하루가 지난 후 project에 기본 log-file을 확인해보니!



20200307에 작성한 logfile이 남아있는 것을 볼 수 있었습니다!!
날짜별로 구분된 logfile확인!

Made by 장경석