

# 스프링 스터디 1주차 (2021.08.15)

- 섹션 1. 프로젝트 환경 설정
- 섹션 2. 스프링 웹 개발 기초



Spring 웹 개발을 위한 IntelliJ 설치 및 환경설정을 진행해보고,  
간단한 웹 개발 기본 동작을 구현해 보았다.

## 라이브러리 살펴보기 → Logging?

Gradle은 의존관계가 있는 라이브러리를 함께 다운로드 한다.

라이브러리 살펴보기 단계에서 Logging에 대한 언급이 나온다.



System.out.print("옛다 Log!"); 로 하면 안됩니다!!

간단하게 Log의 중요성과 어떤 라이브러리가 있는지 살펴보자!!

## Log의 중요성!

## Logging을 하는 이유!

1. 문제 파악
2. 서비스 상태 파악
3. 테스트 검증
4. 빅데이터 활용
  - 대부분의 기업에서 로그를 쌓아 놓고 관리한다!
  - 각 서버들의 서비스 사용 기록(누가, 언제, 어떤 입력값으로, 어떤 서비스를 사용, 어떤 응답을 받았는지, 성공/실패 여부)을 통해 서비스 개선이나 사용량 통계에 활용



Logging은 특히나 클라우드 빅데이터 시대에서 최초 시작점이라고 볼 수 있다.

## Logging Level (in Spring)

- TRACE → DEBUG 보다 아래 단계, 덜 중요하지만 변수를 쫓는 정도의 로그를 찍는데 사용
- DEBUG → DEBUG를 위해서 사용
- INFO → 진행정보, 상태 정보를 로깅
- WARN → 잠재적 오류, 경고성 정보를 로깅
- ERROR → 오류가 발생했을 경우 로깅

보통 INFO는 운영단계, DEBUG는 개발 단계에서 활용한다.

단계 별로 로그의 활용을 다르게 할 수 있다!

ex) 실제 배포 되었을 때 Debug는 활용하지 않게

## Logback

- Ceki Gülcü가 만든 log4j (Log for Java) 프로젝트의 후속작! (log4j 대비 속도 10배 상승!)
- logback-core, logback-classic 및 logback-access의 세 가지 모듈로 나뉜다.
- logback-core는 logback-classic과 logback-access의 기본 토대
- logback-classic은 기본적으로 SLF4J API를 구현하므로 log4j 또는 java.util.logging(JUL)과 같은 기타 로깅 프레임워크와 logback 간에 쉽게 전환할 수 있다. (최신 log4j와 호환가능)

- logback-access 모듈은 HTTP 액세스 로그 기능을 제공하기 위해 Tomcat 및 Jetty와 같은 Servlet 컨테이너와 통합된다. (HTTP 디버깅 기능!)
- SLF4J를 지원하기 때문에 마음에 들지 않으면 언제든지 다른 로거로 Switching이 가능!

**ex)** 자바에서 **Exception**이 발생을 하면 **Stack Trace**를 출력하는데, 자바는 **Exception** 정의가 잘되어 있어서 쉽게 디버깅을 할 수 있다.

하지만 제일 디버깅이 힘든 것 중 하나가 **라이브러리에 의한 Exception** 인데, 이럴 경우 **LOGBack**은 Exception 발생 시 참조했던 외부 라이브러리 버전을 출력하게 해준다.

```
18.835 [btcpool0-7] INFO c.q.l.demo.prime.PrimeAction - 99 is not a valid value
    at org.apache.struts.action.RequestProcessor.processActionPerform(RequestProcessor.java:431) [struts-1.2.9.jar:1.2.9]
    at org.apache.struts.action.RequestProcessor.process(RequestProcessor.java:236) [struts-1.2.9.jar:1.2.9]
    at org.apache.struts.action.ActionServlet.doPost(ActionServlet.java:432) [struts-1.2.9.jar:1.2.9]
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:820) [servlet-api-2.5-6.1.12.jar:6.1.12]
    at org.mortbay.jetty.servlet.ServletHolder.handle(ServletHolder.java:502) [jetty-6.1.12.jar:6.1.12]
    at org.qos.logback.demo.UserServletFilter.doFilter(UserServletFilter.java:44) [classes/:na]
    at org.mortbay.jetty.servlet.ServletHandler$CachedChain.doFilter(ServletHandler.java:1115) [jetty-6.1.12.jar:6.1.12]
    at org.mortbay.jetty.servlet.ServletHandler.handle(ServletHandler.java:361) [jetty-6.1.12.jar:6.1.12]
    at org.mortbay.jetty.webapp.WebAppContext.handle(WebAppContext.java:417) [jetty-6.1.12.jar:6.1.12]
    at org.mortbay.jetty.handler.ContextHandlerCollection.handle(ContextHandlerCollection.java:230) [jetty-6.1.12.jar:6.1.12]
```

## Ref.

### logback 사용해야 하는 이유 (Reasons to prefer logback over log4j)

최근 프로젝트를 진행하면서 자바 로깅 구현체(logger)로 "LOGBack"을 사용하고 있습니다. "개발자의 열리"적 성향이기 보다는 "log4j"에서 제공을 하지 않는 기능 외에 다양한 이점이 있기 때문입니다. 아시는 분들도 있지만, 모르시는

☞ <https://beyondj2ee.wordpress.com/2012/11/09/logback-%EC%82%A%EC%9A%A9%ED%95%B4%EC%95%BC-%ED%95%98%EB%8A%94-%EC%9D%B4%EC%9C%A0-reasons-to-prefer-logback-over-log4j/>



### Logback Home

Logback is intended as a successor to the popular log4j project, picking up where log4j leaves off. Logback's architecture is sufficiently generic so as to apply under different circumstances. At present time, logback is divided into three modules, logback-core, logback-classic and logback-access. The logback-core module lays

☞ <http://logback.qos.ch/>

## SLF4J (Simple Logging Facade for Java)

- 다양한 로깅 프레임워크(java.util.logging, logback, log4j)에 대한 추상화 역할, 사용자가 배포 시 원하는 로깅 프레임워크를 연결할 수 있도록 한다.

- Java로 치면 'Interface 덩어리'
- SLF4J는 Compile 시 하나의 logging framework와 binding 해준다.

## 사용 예시

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class HelloWorld {
    public static void main(String[] args) {
        Logger logger = LoggerFactory.getLogger(HelloWorld.class);
        logger.info("Hello World");
    }
}
```

- 로그 출력

0 [main] INFO HelloWorld - Hello World

## Ref.

### SLF4J

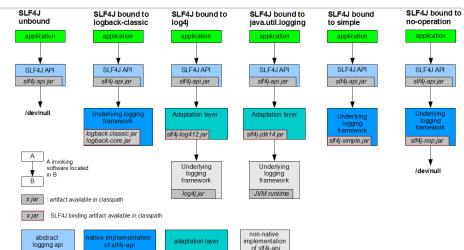
The Simple Logging Facade for Java (SLF4J) serves as a simple facade or abstraction for various logging frameworks (e.g. java.util.logging, logback, log4j) allowing the end user to plug in the desired logging framework at deployment time. Before you start using SLF4J, we highly recommend that you read the two-

 <http://www.slf4j.org/>

### SLF4J Manual

The Simple Logging Facade for Java (SLF4J) serves as a simple facade or abstraction for various logging frameworks, such as java.util.logging, logback and log4j. SLF4J allows the end-user to

 <http://www.slf4j.org/manual.html>



### [SLF4J] slf4j 알고가기 + Logback

SLF4J는 다양한 Logging Framework(java.util.logging, logback, log4j)의 추상화를 제공합니다. SLF4J는 Compile 시 하나의 logging framework와 binding 해줍니다. 가장 큰 것은 다양한 Logging

 <https://hello-bryan.tistory.com/331>

