

스프링 부트 개념과 활용

이 강좌는 스프링부트의 핵심 원리에 대해 설명합니다. "어떻게 `pom.xml`에 이것만 등록했는데 이렇게 많은 의존성이 들어온거지?" 또는 "스프링 부트가 서버인가? 어떻게 웹 애플리케이션이 동작하고 있는거지?" 등이 궁금하셨던 분이라면 본 강좌의 "스프링 부트 원리" 파트에서 그 궁금증을 해결 할 수 있습니다.

다음으로, 스프링 부트가 제공하는 여러 기능을 '핵심 기능'과 '기술 연동'으로 나누어 설명합니다. '핵심 기능'에 해당하는 '**SpringApplication**', '외부 설정' 그리고 '로깅' 등의 기능은 어떠한 기술과 연동하더라도 스프링 부트 애플리케이션의 기반이 되는 기능입니다.

'기술 연동' 부분은 여러분이 만들려는 애플리케이션에 따라 달라집니다. 스프링 부트가 지원하는 기술은 방대하여 본 강좌에서 모든 기술을 다루기는 어렵습니다. 따라서 이 강좌는 주로 웹 **MVC**, 데이터 연동, 시큐리티 그리고 **REST API** 클라이언트 사용법을 다룹니다.

마지막으로 스프링 부트 애플리케이션을 운영 환경에 배포했을 때 유용하게 사용할 수 있는 툴과 기능에 대해 설명합니다.

학습 목표

- 스프링 부트의 핵심 원리를 이해합니다.
- 스프링 부트가 제공하는 주요 기능을 사용할 수 있습니다.
- 스프링 부트를 사용하여 웹 애플리케이션을 개발할 수 있습니다.
- 스프링 부트를 사용하여 여러 데이터 기술과 연동하는 애플리케이션을 개발할 수 있습니다.
- 스프링 부트 애플리케이션의 운영 정보를 관리하고 모니터링 할 수 있습니다.

이번 강의를 통해 여러분은 스프링 부트의 다음과 같은 주제에 대해 학습할 수 있습니다.

- 스프링 부트 원리
 - 의존성 관리
 - 자동 설정
 - 내장 웹 서버
 - 독립적으로 실행 가능한 **JAR**
- 스프링 부트 활용
 - 스프링 부트 핵심 기능
 - **SpringApplication**
 - 외부 설정
 - 로깅
 - 테스트
 - **Spring-Boot-Devtools**
 - 각종 기술 연동
 - 스프링 웹 **MVC**
 - 스프링 데이터
 - 스프링 시큐리티

■ REST 클라이언트

- 스프링 부트 운영
 - 엔드포인트
 - 메트릭스
 - 모니터링

1부: 소개

1. 강의 소개

첫 페이지 참고

2. 강사 소개

백기선

- 현재 마이크로소프트 미국 본사에 근무 중. (그전에는 네이버와 아마존에서 일을 했습니다.)
- 2007년부터 개발자로 일했으며 이제 막 경력 10년이 조금 넘었네요.
- 자바, 스프링 프레임워크, JPA, 하이버네이트를 주로 공부하고 공유해 왔습니다.
- Youtube/백기선 채널에서 코딩 관련 정보를 영상으로 공유하고 있습니다.
- (예전에는 Whiteship.me 라는 블로그에 글도 많이 올렸지만 요즘은 잘 안써요.)
- (더 예전에는 책도 쓰고 번역도 하고 발표도 많이 했었지만 역시나.. 요즘은 안합니다.)

2부: 스프링 부트 시작하기

본격적인 학습에 앞서 본 강의와 스프링 부트를 소개하고 스프링 부트 애플리케이션 만드는 방법을 살펴봅니다.

3. 스프링 부트 소개

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#getting-started-introducing-spring-boot>

- Provide a radically faster and widely accessible getting-started experience for all Spring development.
- Be opinionated out of the box but get out of the way quickly as requirements start to diverge from the defaults.
- Provide a range of non-functional features that are common to large classes of projects (such as embedded servers, security, metrics, health checks, and externalized configuration).
- Absolutely no code generation and no requirement for XML configuration.

4. 스프링 부트 시작하기

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#getting-started-maven-installation>

메이븐 pom.xml에 parent, dependency, plugin 설정.

```
<?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>

  <groupId>com.example</groupId>
  <artifactId>myproject</artifactId>
  <version>0.0.1-SNAPSHOT</version>


  <!-- Inherit defaults from Spring Boot -->
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.0.RELEASE</version>
  </parent>

  <!-- Add typical dependencies for a web application -->
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
  </dependencies>

  <!-- Package as an executable jar -->
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

5. 스프링 부트 프로젝트 생성기

<https://start.spring.io/>

 **Spring Initializr**
Bootstrap your application

Project

Language

Spring Boot

Project Metadata

Dependencies

Maven Project

Gradle Project

Java

Kotlin

Groovy

2.2.1 (SNAPSHOT)

2.2.0

2.1.10 (SNAPSHOT)

2.1.9

Group

com.example

Artifact

demo

> Options

Q




≡

Search dependencies to add

Web, Security, JPA, Actuator, Devtools...

Selected dependencies

No dependency selected

● Light UI  Github  Twitter  Help ▾

6. 스프링 부트 프로젝트 구조

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#using-boot-structuring-our-code>

메이븐 기본 프로젝트 구조와 동일

- 소스 코드 (src\main\java)
- 소스 리소스 (src\main\resource)
- 테스트 코드 (src\test\java)
- 테스트 리소스 (src\test\resource)

메인 애플리케이션 위치

- 기본 패키지

3부: 스프링 부트 원리

2부에서는 스프링 부트의 핵심 원리를 학습합니다. 스프링 부트가 제공하는 '의존성 관리', '자동 설정' 그리고 '내장 서블릿 컨테이너'에 대해 학습합니다.

7. 의존성 관리 이해

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#using-boot-dependency-management>

Spring-Boot-Dependencies

8. 의존성 관리 응용

- 버전 관리 해주는 의존성 추가
- 버전 관리 안해주는 의존성 추가
- 기존 의존성 버전 변경하기
- <https://mvnrepository.com/>

9. 자동 설정 이해

- `@EnableAutoConfiguration` (`@SpringBootApplication` 안에 숨어 있음)
- 빈은 사실 두 단계로 나뉘서 읽힘
 - 1단계: `@ComponentScan`
 - 2단계: `@EnableAutoConfiguration`
- `@ComponentScan`
 - `@Component`
 - `@Configuration` `@Repository` `@Service` `@Controller` `@RestController`
- `@EnableAutoConfiguration`
 - `spring.factories`
 - `org.springframework.boot.autoconfigure.EnableAutoConfiguration`
 - `@Configuration`
 - `@ConditionalOnXxxYyyZzz`

10. 자동 설정 만들기 1부: Starter와 Autoconfigure

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-developing-auto-configuration>

- Xxx-Spring-Boot-Autoconfigure 모듈: 자동 설정
- Xxx-Spring-Boot-Starter 모듈: 필요한 의존성 정의
- 그냥 하나로 만들고 싶을 때는?
 - Xxx-Spring-Boot-Starter
- 구현 방법

1. 의존성 추가

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-autoconfigure</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-autoconfigure-processor</artifactId>
    <optional>true</optional>
  </dependency>
</dependencies>

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-dependencies</artifactId>
      <version>2.0.3.RELEASE</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

2. @Configuration 파일 작성
3. src/main/resource/META-INF에 spring.factories 파일 만들기
4. spring.factories 안에 자동 설정 파일 추가

```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
FQCN,\
FQCN
```

5. mvn install

11. 자동 설정 만들기 2부: @ConfigurationProperties

- 덮어쓰기 방지하기
 - @ConditionalOnMissingBean
- 빈 재정의 수고 덜기
 - @ConfigurationProperties("holoman")
 - @EnableConfigurationProperties(HolomanProperties)
 - 프로퍼티 키값 자동 완성

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-configuration-processor</artifactId>  
  <optional>true</optional>  
</dependency>
```

12. 내장 웹 서버 이해

- 스프링 부트는 서버가 아니다.
 - 톰캣 객체 생성
 - 포트 설정
 - 톰캣에 컨텍스트 추가
 - 서블릿 만들기
 - 톰캣에 서블릿 추가
 - 컨텍스트에 서블릿 맵핑
 - 톰캣 실행 및 대기
- 이 모든 과정을 보다 상세히 또 유연하고 설정하고 실행해주는게 바로 스프링 부트의 자동 설정.
 - `ServletWebServerFactoryAutoConfiguration` (서블릿 웹 서버 생성)
 - `TomcatServletWebServerFactoryCustomizer` (서버 커스터마이징)
 - `DispatcherServletAutoConfiguration`
 - 서블릿 만들고 등록

13. 내장 웹 서버 응용 1부: 컨테이너와 서버 포트

<https://docs.spring.io/spring-boot/docs/current/reference/html/howto-embedded-web-servers.html>

- 다른 서블릿 컨테이너로 변경
- 웹 서버 사용 하지 않기
- 포트
 - `server.port`
 - 랜덤 포트
 - `ApplicationListner<ServletWebServerInitializedEvent>`

14. 내장 웹 서버 응용 2부: HTTPS와 HTTP2

<https://opentutorials.org/course/228/4894>

<https://gist.github.com/keesun/f93f0b83d7232137283450e08a53c4fd>

- HTTPS 설정하기
 - 키스토어 만들기
 - HTTP는 못쓰네?
- HTTP 커넥터는 코딩으로 설정하기
 - <https://github.com/spring-projects/spring-boot/tree/v2.0.3.RELEASE/spring-boot-samples/spring-boot-sample-tomcat-multi-connectors>
- HTTP2 설정
 - `server.http2.enable`
 - 사용하는 서블릿 컨테이너 마다 다름.

15. 톰캣 HTTP2

- JDK9와 Tomcat 9+ 추천
- 그 이하는 아래 링크 참고

<https://docs.spring.io/spring-boot/docs/current/reference/html/howto-embedded-web-servers.html#howto-configure-http2-tomcat>

16. 독립적으로 실행 가능한 JAR

<https://docs.spring.io/spring-boot/docs/current/reference/html/executable-jar.html>

“그리고 보니 JAR 파일 하나로 실행할 수 있네?”

- mvn package를 하면 실행 가능한 **JAR** 파일 “하나가” 생성 됨.
- spring-maven-plugin이 해주는 일 (패키징)
- 과거 “uber” jar 를 사용
 - 모든 클래스 (의존성 및 애플리케이션)를 하나로 압축하는 방법
 - 뭐가 어디에서 온건지 알 수가 없음
 - 무슨 라이브러리를 쓰는건지..
 - 내용은 다르지만 이름이 같은 파일은 또 어떻게?
- 스프링 부트의 전략
 - 내장 JAR : 기본적으로 자바에는 내장 JAR를 로딩하는 표준적인 방법이 없음.
 - 애플리케이션 클래스와 라이브러리 위치 구분
 - org.springframework.boot.loader.jar.JarFile을 사용해서 내장 JAR를 읽는다.
 - org.springframework.boot.loader.Launcher를 사용해서 실행한다.

17. 스프링 부트 원리 정리

- 의존성 관리
 - 이것만 넣어도 이만큼이나 다 알아서 가져오네?
- 자동 설정
 - `@EnableAutoConfiguration`이 뭘 해주는지 알겠어.
- 내장 웹 서버
 - 아 스프링 부트가 서버가 아니라 내장 서버를 실행하는 거군.
- 독립적으로 실행 가능한 **JAR**
 - `spring-boot-maven` 플러그인이 이런걸 해주는구나..

4부: 스프링 부트 활용

3부에서는 스프링 부트가 제공하는 여러 기능을 사용하며 원하는대로 커스터마이징 하는 방법을 학습합니다.

18. 스프링 부트 활용 소개

스프링 부트 핵심 기능	각종 기술 연동
<ul style="list-style-type: none">• SpringApplication• 외부 설정• 프로파일• 로깅• 테스트• Spring-Dev-Tools	<ul style="list-style-type: none">• 스프링 웹 MVC• 스프링 데이터• 스프링 시큐리티• REST API 클라이언트• 다루지 않은 내용들

19. SpringApplication 1부

<https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-spring-application.html#boot-features-spring-application>

- 기본 로그 레벨 INFO
 - 뒤에 로깅 수업때 자세히 살펴볼 예정
- FailureAnalyzer
- 배너
 - banner.txt | gif | jpg | png
 - classpath 또는 spring.banner.location
 - \${spring-boot.version} 등의 변수를 사용할 수 있음.
 - Banner 클래스 구현하고 SpringApplication.setBanner()로 설정 가능.
 - 배너 끄는 방법
- SpringApplicationBuilder로 빌더 패턴 사용 가능

20. SpringApplication 2부

<https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-spring-application.html#boot-features-application-events-and-listeners>

- ApplicationEvent 등록
 - ApplicationContext를 만들기 전에 사용하는 리스너는 @Bean으로 등록할 수 없다.
 - SpringApplication.addListners()
- WebApplicationType 설정
- 애플리케이션 아규먼트 사용하기
 - ApplicationArguments를 빈으로 등록해 주니까 가져다 쓰면 됨.
- 애플리케이션 실행한 뒤 뭔가 실행하고 싶을 때
 - ApplicationRunner (추천) 또는 CommandLineRunner
 - 순서 지정 가능 @Order

21. 외부 설정 1부

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-external-config>

사용할 수 있는 외부 설정

- properties
- YAML
- 환경 변수
- 커맨드 라인 아규먼트

프로퍼티 우선 순위

1. 유저 홈 디렉토리에 있는 `spring-boot-dev-tools.properties`
2. 테스트에 있는 `@TestPropertySource`
3. `@SpringBootTest` 애노테이션의 `properties` 애트리뷰트
4. 커맨드 라인 아규먼트
5. **`SPRING_APPLICATION_JSON` (환경 변수 또는 시스템 프로티) 에 들어있는 프로퍼티**
6. `ServletConfig` 파라미터
7. `ServletContext` 파라미터
8. `java:comp/env JNDI` 애트리뷰트
9. `System.getProperties()` 자바 시스템 프로퍼티
10. **OS** 환경 변수
11. `RandomValuePropertySource`
12. **JAR** 밖에 있는 특정 프로파일용 **application properties**
13. **JAR** 안에 있는 특정 프로파일용 **application properties**
14. **JAR** 밖에 있는 **application properties**
15. **JAR** 안에 있는 **application properties**
16. `@PropertySource`
17. 기본 프로퍼티 (`SpringApplication.setDefaultProperties`)

`application.properties` 우선 순위 (높은게 낮은걸 덮어 씁니다.)

1. `file:./config/`
2. `file:./`
3. `classpath:/config/`
4. `classpath:/`

랜덤값 설정하기

- `${random.*}`

플레이스 홀더

- `name = keesun`
- `fullName = ${name} baik`

22. 외부 설정 2부

타입-세이프 프로퍼티 **@ConfigurationProperties**

- 여러 프로퍼티를 묶어서 읽어올 수 있음
- 빈으로 등록해서 다른 빈에 주입할 수 있음
 - **@EnableConfigurationProperties**
 - **@Component**
 - **@Bean**
- 융통성 있는 바인딩
 - context-path (케밥)
 - context_path (언더스코어)
 - contextPath (캐멀)
 - CONTEXTPATH
- 프로퍼티 타입 컨버전
 - **@DurationUnit**
- 프로퍼티 값 검증
 - **@Validated**
 - JSR-303 (**@NotNull**, ...)
- 메타 정보 생성
- **@Value**
 - SpEL 을 사용할 수 있지만...
 - 위에 있는 기능들은 전부 사용 못합니다.

23. 프로파일

@Profile 애노테이션은 어디에?

- @Configuration
- @Component

어떤 프로파일을 활성화 할 것인가?

- spring.profiles.active

어떤 프로파일을 추가할 것인가?

- spring.profiles.include

프로파일용 프로퍼티

- application-{profile}.properties

24. 로깅 1부: 스프링 부트 기본 로거 설정

로깅 퍼사드 VS 로거

- 로깅 퍼사드: Commons Logging, **SLF4j**
- 로거: JUL, Log4J2, **Logback**

스프링 5에 로거 관련 변경 사항

- <https://docs.spring.io/spring/docs/5.0.0.RC3/spring-framework-reference/overview.html#overview-logging>
- Spring-JCL
 - Commons Logging -> SLF4j or Log4j2
 - pom.xml에 exclusion 안해도 됨.

스프링 부트 로깅

- 기본 포맷
- `--debug` (일부 핵심 라이브러리만 디버깅 모드로)
- `--trace` (전부 다 디버깅 모드로)
- 컬러 출력: `spring.output.ansi.enabled`
- 파일 출력: `logging.file` 또는 `logging.path`
- 로그 레벨 조정: `logging.level.패지키` = 로그 레벨

25. 로깅 2부: 커스터마이징

<https://docs.spring.io/spring-boot/docs/current/reference/html/howto-logging.html>

커스텀 로그 설정 파일 사용하기

- Logback: logback-spring.xml
- Log4J2: log4j2-spring.xml
- JUL (비추): logging.properties
- Logback extension
 - 프로파일 <springProfile name="프로파일">
 - Environment 프로퍼티 <springProperty>

로거를 Log4j2로 변경하기

- <https://docs.spring.io/spring-boot/docs/current/reference/html/howto-logging.html#howto-configure-log4j-for-logging>

26. 테스트

시작은 일단 `spring-boot-starter-test`를 추가하는 것 부터

- `test` 스코프로 추가.

@SpringBootTest

- `@RunWith(SpringRunner.class)`랑 같이 써야 함. (JUnit 4 기반에서만)
- 빈 설정 파일은 설정을 안해주나? 알아서 찾습니다. (`@SpringBootTest`)
- `webEnvironment`
 - `MOCK`: mock servlet environment. 내장 톰캣 구동 안 함.
 - `RANDOM_PORT`, `DEFINED_PORT`: 내장 톰캣 사용 함.
 - `NONE`: 서블릿 환경 제공 안 함.

@MockBean

- `ApplicationContext`에 들어있는 빈을 `Mock`으로 만든 객체로 교체 함.
- 모든 `@Test` 마다 자동으로 리셋.

슬라이스 테스트

- 레이어 별로 잘라서 테스트하고 싶을 때
- `@JsonTest`
- `@WebMvcTest`
- `@WebFluxTest`
- `@DataJpaTest`
- ...

27. 테스트 유틸

엇? 스프링 부트 테스트 끝난줄 알았는데...

- **OutputCapture**
- TestPropertyValues
- TestRestTemplate
- ConfigFileApplicationContextInitializer

28. Spring-Boot-Devtools

- 캐시 설정을 개발 환경에 맞게 변경.
- 클래스패스에 있는 파일이 변경 될 때마다 자동으로 재시작.
 - 직접 껐다 켜는거 (**cold starts**)보다 빠르다. 왜?
 - 릴로딩 보다는 느리다. (JRebel 같은건 아님)
 - 리스타트 하고 싶지 않은 리소스는? `spring.devtools.restart.exclude`
 - 리스타트 기능 끄려면? `spring.devtools.restart.enabled = false`
- 라이브 릴로드? 리스타트 했을 때 브라우저 자동 리프레시 하는 기능
 - 브라우저 플러그인 설치해야 함.
 - 라이브 릴로드 서버 끄려면? `spring.devtools.liveload.enabled = false`
- 글로벌 설정
 - `~/.spring-boot-devtools.properties`
- 리모트 애플리케이션

29. 스프링 웹 MVC 1부: 소개

- 스프링 웹 MVC
 - <https://docs.spring.io/spring/docs/5.0.7.RELEASE/spring-framework-reference/web.html#spring-web>
- 스프링 부트 MVC
 - 자동 설정으로 제공하는 여러 기본 기능 (앞으로 살펴볼 예정)
- 스프링 MVC 확장
 - @Configuration + WebMvcConfigurer
- 스프링 MVC 재정의
 - @Configuration + @EnableWebMvc

30. 스프링 웹 MVC 2부: HttpMessageConverters

<https://docs.spring.io/spring/docs/5.0.7.RELEASE/spring-framework-reference/web.html#mvc-config-message-converters>

HTTP 요청 본문을 객체로 변경하거나, 객체를 HTTP 응답 본문으로 변경할 때 사용.

{“username”:”keesun”, “password”:”123”} <-> User

- @RequestBody
- @ResponseBody

31. 스프링 웹 MVC 3부: ViewResolver

스프링 부트

- 뷰 리졸버 설정 제공
- `HttpMessageConvertersAutoConfiguration`

XML 메시지 컨버터 추가하기

```
<dependency>  
  <groupId>com.fasterxml.jackson.dataformat</groupId>  
  <artifactId>jackson-dataformat-xml</artifactId>  
  <version>2.9.6</version>  
</dependency>
```


32. 스프링 웹 MVC 4부: 정적 리소스 지원

정적 리소스 �핑 “/**”

- 기본 리소스 위치
 - classpath:/static
 - classpath:/public
 - classpath:/resources/
 - classpath:/META-INF/resources
 - 예) “/hello.html” => /static/hello.html
 - spring.mvc.static-path-pattern: 맵핑 설정 변경 가능
 - spring.mvc.static-locations: 리소스 찾을 위치 변경 가능
- Last-Modified 헤더를 보고 304 응답을 보냄.
- ResourceHttpRequestHandler가 처리함.
 - WebMvcConfigurer의 addResourceHandlers로 커스터마이징 할 수 있음

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
    registry.addResourceHandler("/m/**")
        .addResourceLocations("classpath:/m/")
        .setCachePeriod(20);
}
```

33. 스프링 웹 MVC 5부: 웹JAR

웹JAR 맵핑 “/webjars/**”

- 버전 생략하고 사용하려면
 - webjars-locator-core 의존성 추가

```
<script src="/webjars/jquery/dist/jquery.min.js"></script>
<script>
    $(function() {
        console.log("ready!");
    });
</script>
```

34. 스프링 웹 MVC 6부: index 페이지와 파비콘

웰컴 페이지

- index.html 찾아 보고 있으면 제공.
- index.템플릿 찾아 보고 있으면 제공.
- 둘 다 없으면 에러 페이지.

파비콘

- favicon.ico
- 파이콘 만들기 <https://favicon.io/>
- 파비콘이 안 바뀔 때?
 - <https://stackoverflow.com/questions/2208933/how-do-i-force-a-favicon-refresh>

35. 스프링 웹 MVC 7부: Thymeleaf

스프링 부트가 자동 설정을 지원하는 템플릿 엔진

- FreeMarker
- Groovy
- **Thymeleaf**
- Mustache

JSP를 권장하지 않는 이유

- JAR 패키징 할 때는 동작하지 않고, WAR 패키징 해야 함.
- Undertow는 JSP를 지원하지 않음.
- <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-jsp-limitations>

Thymeleaf 사용하기

- <https://www.thymeleaf.org/>
- <https://www.thymeleaf.org/doc/articles/standarddialect5minutes.html>
- 의존성 추가: spring-boot-starter-thymeleaf
- 템플릿 파일 위치: /src/main/resources/**template/**
- 예제:
<https://github.com/thymeleaf/thymeleafexamples-stsm/blob/3.0-master/src/main/webapp/WEB-INF/templates/seedstartermng.html>

36. 스프링 웹 MVC 8부: HtmlUnit

HTML 템플릿 뷰 테스트를 보다 전문적으로 하자.

- <http://htmlunit.sourceforge.net/>
- <http://htmlunit.sourceforge.net/gettingStarted.html>
- 의존성 추가

```
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>htmlunit-driver</artifactId>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>net.sourceforge.htmlunit</groupId>
  <artifactId>htmlunit</artifactId>
  <scope>test</scope>
</dependency>
```

- @Autowire WebClient

37. 스프링 웹 MVC 9부: ExceptionHandler

스프링 @MVC 예외 처리 방법

- @ControllerAdvice
- @ExceptionHandler

스프링 부트가 제공하는 기본 예외 처리기

- BasicExceptionHandler
 - HTML과 JSON 응답 지원
- 커스터마이징 방법
 - ErrorController 구현

커스텀 에러 페이지

- 상태 코드 값에 따라 에러 페이지 보여주기
- src/main/resources/static/template/error/
- 404.html
- 5xx.html
- ErrorViewResolver 구현

38. 스프링 웹 MVC 10부: Spring HATEOAS

Hypermedia As The Engine Of Application State

- 서버: 현재 리소스와 연관된 링크 정보를 클라이언트에게 제공한다.
- 클라이언트: 연관된 링크 정보를 바탕으로 리소스에 접근한다.
- 연관된 링크 정보
 - **Relation**
 - **Hypertext Reference**)
- spring-boot-starter-hateoas 의존성 추가
- <https://spring.io/understanding/HATEOAS>
- <https://spring.io/guides/gs/rest-hateoas/>
- <https://docs.spring.io/spring-hateoas/docs/current/reference/html/>

ObjectMapper 제공

- spring.jackson.*
- Jackson2ObjectMapperBuilder

LinkDiscoverers 제공

- 클라이언트 쪽에서 링크 정보를 Rel 이름으로 찾을때 사용할 수 있는 XPath 확장 클래스

39. 스프링 웹 MVC 11부: CORS

SOP과 CORS

- Single-Origin Policy
- Cross-Origin Resource Sharing
- Origin?
 - URI 스키마 (http, https)
 - hostname (whiteship.me, localhost)
 - 포트 (8080, 18080)

스프링 MVC @CrossOrigin

- <https://docs.spring.io/spring/docs/5.0.7.RELEASE/spring-framework-reference/web.html#mvc-cors>
- @Controller나 @RequestMapping에 추가하거나
- WebMvcConfigurer 사용해서 글로벌 설정

40. 스프링 데이터 1부: 소개

SQL DB	NoSQL
<ul style="list-style-type: none">• 인메모리 데이터베이스 지원• DataSource 설정• DBCP 설정• JDBC 사용하기• 스프링 데이터 JPA 사용하기• jOOQ 사용하기• 데이터베이스 초기화• 데이터베이스 마이그레이션 툴 연동하기	<ul style="list-style-type: none">• Redis (Key/Value)• MongoDB (Document)• Neo4J (Graph)• Gemfire (IMDG)• Solr (Search)• Elasticsearch (Search & Analytics)• Cassandra• Couchbase• LDAP• InfluxDB

41. 스프링 데이터 2부: 인메모리 데이터베이스

지원하는 인-메모리 데이터베이스

- **H2 (추천, 콘솔 때문에...)**
- HSQL
- Derby

Spring-JDBC가 클래스패스에 있으면 자동 설정이 필요한 빈을 설정 해줍니다.

- DataSource
- JdbcTemplate

인-메모리 데이터베이스 기본 연결 정보 확인하는 방법

- URL: "testdb"
- username: "sa"
- password: ""

H2 콘솔 사용하는 방법

- spring-boot-devtools를 추가하거나...
- spring.h2.console.enabled=true 만 추가.
- /h2-console로 접속 (이 path도 바꿀 수 있음)

실습 코드

- CREATE TABLE USER (ID INTEGER NOT NULL, name VARCHAR(255), PRIMARY KEY (id))
- INSERT INTO USER VALUES (1, 'keesun')

42. 스프링 데이터 3부: MySQL

지원하는 DBCP

1. [HikariCP](#) (기본)
 - <https://github.com/brettwooldridge/HikariCP#frequently-used>
2. [Tomcat CP](#)
3. [Commons DBCP2](#)

DBCP 설정

- `spring.datasource.hikari.*`
- `spring.datasource.tomcat.*`
- `spring.datasource.dbcp2.*`

MySQL 커넥터 의존성 추가

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>
```

MySQL 추가 (도커 사용)

- `docker run -p 3306:3306 --name mysql_boot -e MYSQL_ROOT_PASSWORD=1 -e MYSQL_DATABASE=springboot -e MYSQL_USER=keesun -e MYSQL_PASSWORD=pass -d mysql`
- `docker exec -i -t mysql_boot bash`
- `mysql -u root -p`

MySQL용 Datasource 설정

- `spring.datasource.url=jdbc:mysql://localhost:3306/springboot?useSSL=false`
- `spring.datasource.username=keesun`
- `spring.datasource.password=pass`

MySQL 접속시 에러

MySQL 5.* 최신 버전 사용할 때

문제	Sat Jul 21 11:17:59 PDT 2018 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false' . You need either to explicitly disable SSL by setting useSSL=false , or set useSSL=true and provide truststore for server certificate verification.
----	---

해결	jdbc:mysql://localhost:3306/springboot? useSSL=false
----	---

MySQL 8.* 최신 버전 사용할 때

문제	com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionException : Public Key Retrieval is not allowed
해결	jdbc:mysql://localhost:3306/springboot?useSSL=false& allowPublicKeyRetrieval=true

MySQL 라이선스 (GPL) 주의

- MySQL 대신 MariaDB 사용 검토
- 소스 코드 공개 의무 여부 확인

43. 스프링 데이터 4부: PostgreSQL

의존성 추가

```
<dependency>  
  <groupId>org.postgresql</groupId>  
  <artifactId>postgresql</artifactId>  
</dependency>
```

PostgreSQL 설치 및 서버 실행 (docker)

```
docker run -p 5432:5432 -e POSTGRES_PASSWORD=pass -e  
POSTGRES_USER=keesun -e POSTGRES_DB=springboot --name postgres_boot -d  
postgres
```

```
docker exec -i -t postgres_boot bash
```

```
su - postgres
```

```
psql springboot
```

데이터베이스 조회

```
\list
```

테이블 조회

```
\dt
```

쿼리

```
SELECT * FROM account;
```

PostgreSQL 경고 메시지

경고	org.postgresql.jdbc.PgConnection.createClob() is not yet implemented
해결	spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true

44. 스프링 데이터 5부: 스프링 데이터 JPA

ORM(Object-Relational Mapping)과 JPA (Java Persistence API)

- 객체와 릴레이션을 맵핑할 때 발생하는 개념적 불일치를 해결하는 프레임워크
- <http://hibernate.org/orm/what-is-an-orm/>
- JPA: ORM을 위한 자바 (EE) 표준

스프링 데이터 JPA

- Repository 빈 자동 생성
- 쿼리 메소드 자동 구현
- @EnableJpaRepositories (스프링 부트가 자동으로 설정 해줌.)
- SDJ -> JPA -> Hibernate -> Datasource

45. 스프링 데이터 6부: Spring-Data-JPA 연동

스프링 데이터 JPA 의존성 추가

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

스프링 데이터 JPA 사용하기

- @Entity 클래스 만들기
- Repository 만들기

스프링 데이터 리파지토리 테스트 만들기

- H2 DB를 테스트 의존성에 추가하기
- @DataJpaTest (슬라이스 테스트) 작성

46. 스프링 데이터 7부: 데이터베이스 초기화

JPA를 사용한 데이터베이스 초기화

- `spring.jpa.hibernate.ddl-auto`
- `spring.jpa.generate-dll=true`로 설정 해줘야 동작함.

SQL 스크립트를 사용한 데이터베이스 초기화

- `schema.sql` 또는 `schema-${platform}.sql`
- `data.sql` 또는 `data-${platform}.sql`
- `${platform}` 값은 `spring.datasource.platform` 으로 설정 가능.

47. 스프링 데이터 8부: 데이터베이스 마이그레이션

Flyway와 Liquibase가 대표적인데, 지금은 Flyway를 사용하겠습니다.

<https://docs.spring.io/spring-boot/docs/2.0.3.RELEASE/reference/htmlsingle/#howto-execute-flyway-database-migrations-on-startup>

의존성 추가

- `org.flywaydb:flyway-core`

마이그레이션 디렉토리

- `db/migration` 또는 `db/migration/{vendor}`
- `spring.flyway.locations`로 변경 가능

마이그레이션 파일 이름

- `V숫자__이름.sql`
- `V`는 꼭 대문자로.
- 숫자는 순차적으로 (타임스탬프 권장)
- 숫자와 이름 사이에 언더바 두 개.
- 이름은 가능한 서술적으로.

48. 스프링 데이터 9부: Redis

캐시, 메시지 브로커, 키/밸류 스토어 등으로 사용 가능.

의존성 추가

- `spring-boot-starter-data-redis`

Redis 설치 및 실행 (도커)

- `docker run -p 6379:6379 --name redis_boot -d redis`
- `docker exec -i -t redis_boot redis-cli`

스프링 데이터 Redis

- <https://projects.spring.io/spring-data-redis/>
- `StringRedisTemplate` 또는 `RedisTemplate`
- extends `CrudRepository`

Redis 주요 커맨드

- <https://redis.io/commands>
- `keys *`
- `get {key}`
- `hgetall {key}`
- `hget {key} {column}`

커스터마이징

- `spring.redis.*`

49. 스프링 데이터 10부: MongoDB

[MongoDB](#)는 JSON 기반의 도큐먼트 데이터베이스입니다.

의존성 추가

- spring-boot-starter-data-mongodb

MongoDB 설치 및 실행 (도커)

- docker run -p 27017:27017 --name mongo_boot -d mongo
- docker exec -i -t mongo_boot bash
- mongo

스프링 데이터 몽고DB

- MongoTemplate
- MongoRepository
- 내장형 MongoDB (테스트용)
 - de.flapdoodle.embed:de.flapdoodle.embed.mongo
- @DataMongoTest

50. 스프링 데이터 11부: Neo4j

[Neo4j](#)는 노드간의 연관 관계를 영속화하는데 유리한 그래프 데이터베이스 입니다.

의존성 추가

- `spring-boot-starter-data-neo4j`

Neo4j 설치 및 실행 (도커)

- `docker run -p 7474:7474 -p 7687:7687 -d --name noe4j_boot neo4j`
- <http://localhost:7474/browser>

스프링 데이터 Neo4J

- `Neo4jTemplate` (Deprecated)
- **`SessionFactory`**
- `Neo4jRepository`

51. 스프링 데이터 12부: 정리

<https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#boot-features-sql>

52. 스프링 시큐리티 1부: spring-boot-starter-security

스프링 시큐리티

- 웹 시큐리티
- 메소드 시큐리티
- 다양한 인증 방법 지원
 - LDAP, 폼 인증, Basic 인증, OAuth, ...

스프링 부트 시큐리티 자동 설정

- SecurityAutoConfiguration
- UserDetailsServiceAutoConfiguration
- spring-boot-starter-security
 - 스프링 시큐리티 5.* 의존성 추가
- 모든 요청에 인증이 필요함.
- 기본 사용자 생성
 - Username: user
 - Password: 애플리케이션을 실행할 때 마다 랜덤 값 생성 (콘솔에 출력 됨.)
 - spring.security.user.name
 - spring.security.user.password
- 인증 관련 각종 이벤트 발생
 - DefaultAuthenticationEventPublisher 빈 등록
 - 다양한 인증 에러 핸들러 등록 가능

스프링 부트 시큐리티 테스트

- <https://docs.spring.io/spring-security/site/docs/current/reference/html/test-method.html>
!

53. 스프링 시큐리티 2부: 시큐리티 설정 커스터마이징

1. 웹 시큐리티 설정

```
@Configuration
public class WebSecurityConfig extends WebSecurityConfigurerAdapter
{
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/", "/hello").permitAll()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .and()
            .httpBasic();
    }
}
```

2. UserDetailsServie 구현

<https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/#jc-authentication-userdetailsservice>

3. PasswordEncoder 설정 및 사용

<https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/#core-services-password-encoding>

54. 스프링 REST 클라이언트 1부: RestTemplate과 WebClient

RestTemplate

- Blocking I/O 기반의 Synchronous API
- RestTemplateAutoConfiguration
- 프로젝트에 spring-web 모듈이 있다면 RestTemplate**Builder**를 빈으로 등록해 줍니다.
- <https://docs.spring.io/spring/docs/current/spring-framework-reference/integration.html#rest-client-access>

WebClient

- Non-Blocking I/O 기반의 Asynchronous API
- WebClientAutoConfiguration
- 프로젝트에 spring-webflux 모듈이 있다면 WebClient.**Builder**를 빈으로 등록해 줍니다.
- <https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.html#webflux-client>

55. 스프링 REST 클라이언트 2부: 커스터마이징

RestTemplate

- 기본으로 `java.net.HttpURLConnection` 사용.
- 커스터마이징
 - 로컬 커스터마이징
 - 글로벌 커스터마이징
 - `RestTemplateCustomizer`
 - 빈 재정의

WebClient

- 기본으로 `Reactor Netty`의 HTTP 클라이언트 사용.
- 커스터마이징
 - 로컬 커스터마이징
 - 글로벌 커스터마이징
 - `WebClientCustomizer`
 - 빈 재정의

56. 그밖에 다양한 기술 연동

- 캐시
- 메시징
- **Validation**
- 이메일 전송
- **JTA**
- 스프링 인티그레이션
- 스프링 세션
- **JMX**
- 웹소켓
- 코틀린
- ...

5부: 스프링 부트 운영

스프링 부트는 애플리케이션 운영 환경에서 유용한 기능을 제공합니다. 스프링 부트가 제공하는 엔드포인트와 메트릭스 그리고 그 데이터를 활용하는 모니터링 기능에 대해 학습합니다.

57. 스프링 부트 Actuator 1부: 소개

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#production-ready-endpoints>

의존성 추가

- `spring-boot-starter-actuator`

애플리케이션의 각종 정보를 확인할 수 있는 Endpoints

- 다양한 Endpoints 제공.
- JMX 또는 HTTP를 통해 접근 가능 함.
- shutdown을 제외한 모든 Endpoint는 기본적으로 활성화 상태.
- 활성화 옵션 조정
 - `management.endpoints.enabled-by-default=false`
 - `management.endpoint.info.enabled=true`

58. 스프링 부트 Actuator 2부: JMX와 HTTP

JConsole 사용하기

- <https://docs.oracle.com/javase/tutorial/jmx/mbeans/>
- <https://docs.oracle.com/javase/7/docs/technotes/guides/management/jconsole.html>

VisualVM 사용하기

- <https://visualvm.github.io/download.html>

HTTP 사용하기

- `/actuator`
- `health`와 `info`를 제외한 대부분의 Endpoint가 기본적으로 비공개 상태
- 공개 옵션 조정
 - `management.endpoints.web.exposure.include=*`
 - `management.endpoints.web.exposure.exclude=env,beans`

59. 스프링 부트 Actuator 3부: Spring-Boot-Admin

<https://github.com/codecentric/spring-boot-admin>

스프링 부트 Actuator UI 제공

어드민 서버 설정

```
<dependency>  
  <groupId>de.codecentric</groupId>  
  <artifactId>spring-boot-admin-starter-server</artifactId>  
  <version>2.0.1</version>  
</dependency>
```

@EnableAdminServer

클라이언트 설정

```
<dependency>  
  <groupId>de.codecentric</groupId>  
  <artifactId>spring-boot-admin-starter-client</artifactId>  
  <version>2.0.1</version>  
</dependency>
```

spring.boot.admin.client.url=http://localhost:8080

management.endpoints.web.exposure.include=*

6부: 마무리

60. 스프링 부트 마무리

스프링 부트 원리

- 의존성 관리
- 자동 설정
- 내장 웹 서버
- JAR 패키징

스프링 부트 활용

- 스프링 부트 핵심 기능
- 다양한 기술 연동

스프링 부트 운영

- Actuator
- 스프링 부트 어드민

수강 후기 꼭 부탁드립니다.
질문 & 답변 게시판 많은 이용 부탁드립니다.

감사합니다.