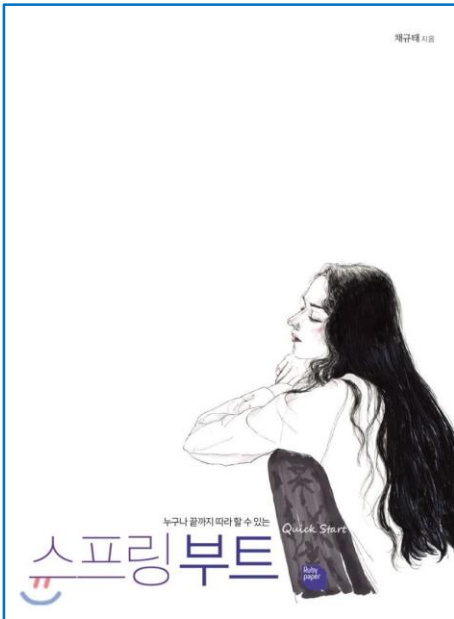


Spring Boot

Bok, Jong Soon
javaexpert@nate.com
<https://github.com/swacademy/Spring5>

Reference

- <https://gist.github.com/ihoneymon/8a905e1dd8393b6b9298>
- <https://www.concretepage.com/spring-boot/spring-boot-thymeleaf-maven-example>
- <https://medium.com/@trevormydata/week-5-thymeleaf-with-spring-mvc-rapid-introduction-to-the-essentials-799f1fba8c07>



Spring Boot

- Spring Framework는 시간이 지나면서 하위 Project가 점점 늘어나서 **방대해졌기 때문**에 그 중에서 무엇을 사용해야 할지 알 수 없다는 의견이 많았다.
- 실제로 각 Project를 조합해서 사용하려면 **초기에 설정할 것도 많고 조합하는 방법 자체에도 Knowhow가 필요**했다.
- 하지만, Spring MVC를 사용하기 위해서는 필요한 Framework들과 Library를 정확하게 설정하지 않으면 안된다.
- 또한, 기본적인 처리를 구축하기 위해 MVC의 각 Code를 작성해야 한다.
- 즉, 실질적인 프로그래밍에 들어가기 전에 하는 작업이 매우 복잡하다.
- 그래서 좀 더 **효율적**으로, **최소한의 작업만**으로 Spring MVC를 사용한 Web Application을 개발할 수 없을까 하는 고민에서 탄생한 것.
- 이때 등장한 것인 Spring Boot이다.

Spring Boot (Cont.)

- 덕분에 개발자는 적은 양의 Application Code를 작성해서 바로 실행할 수 있다.
- 실운영 Server에서 사용할 수 있는 Web Application을 최소한의 작업으로 개발할 수 있게 설계됐다.
- 기존의 수 많은 XML 설정 File을 이용하는 Java EE 개발 방식에서 **설정 File 을 사용하지 않고 Annotation 을 사용**하는 방식으로 방향을 바꾸었다.
- 또한 Java EE의 세계에서는 비슷한 처리를 하는 비슷한 Code를 반복해서 작성하는 경우가 많았지만 이것을 **Code 를 작성하지 않고 처리를 구현**하는 방식으로 변경했다.
- 그래서 실제로 Database 관련 처리는 Class만 정의하면 Method의 정의 없이 구현이 가능하다.
- 결론은 **Spring Boot는 Spring MVC를 대체하는 것이 아니라 Spring MVC를 좀 더 편하게 사용하도록 만들어 주는 도구**이다.

Spring Boot (Cont.)

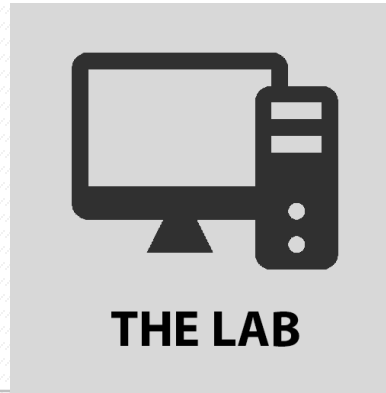
- Spring Boot는 Dropwizard(<https://www.dropwizard.io>)에서 영향을 받아 개발되었다.
- Spring Boot를 이용하는 Application은 Groovy 계열과 Java 계열로 크게 나누어 생각할 수 있다.
- Groovy를 이용한 Application
 - 본격적으로 개발에 들어가기에 앞서 Prototype 등을 빠르게 개발해야 하는 경우에 도움
 - 단시간에 Application의 틀을 만들어서 보여주고 그것을 바탕으로 본격적인 개발에 들어가는 것
- Java를 이용한 Application
 - Maven or Gradle이라는 Build Tool을 사용하는 Project로 생성되며 Java를 사용해 Code한다.

Spring Boot 기능

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated *starter* component to simplify your build configuration
- Automatically configure Spring whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

Spring Boot 시작하기

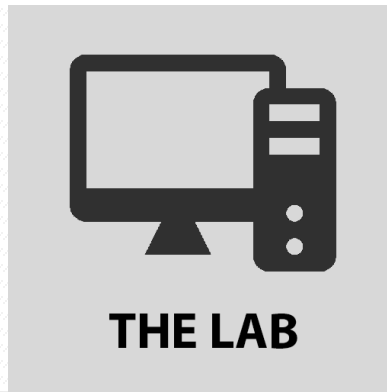
- Maven으로 Template Project 시작하기
- Groovy로 Application 개발하기
- <http://start.spring.io>에서 생성하기
- STS로 Project 만들기



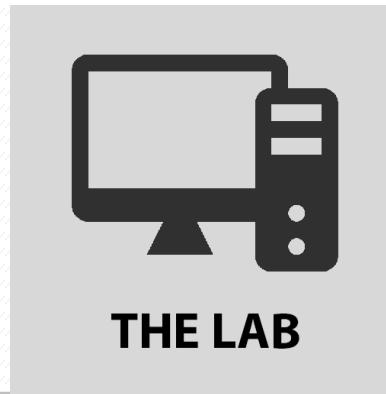
Task 1. Maven으로 Spring Boot Project 생성하기



Task 2. STS로 Spring Boot Application 개발하기

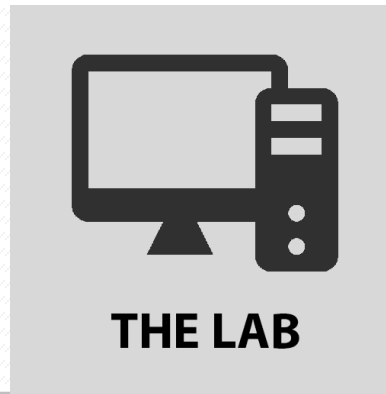


Task 3. Groovy로 Application 개발하기



Task 4. SPRING INITIALIZR(Maven)



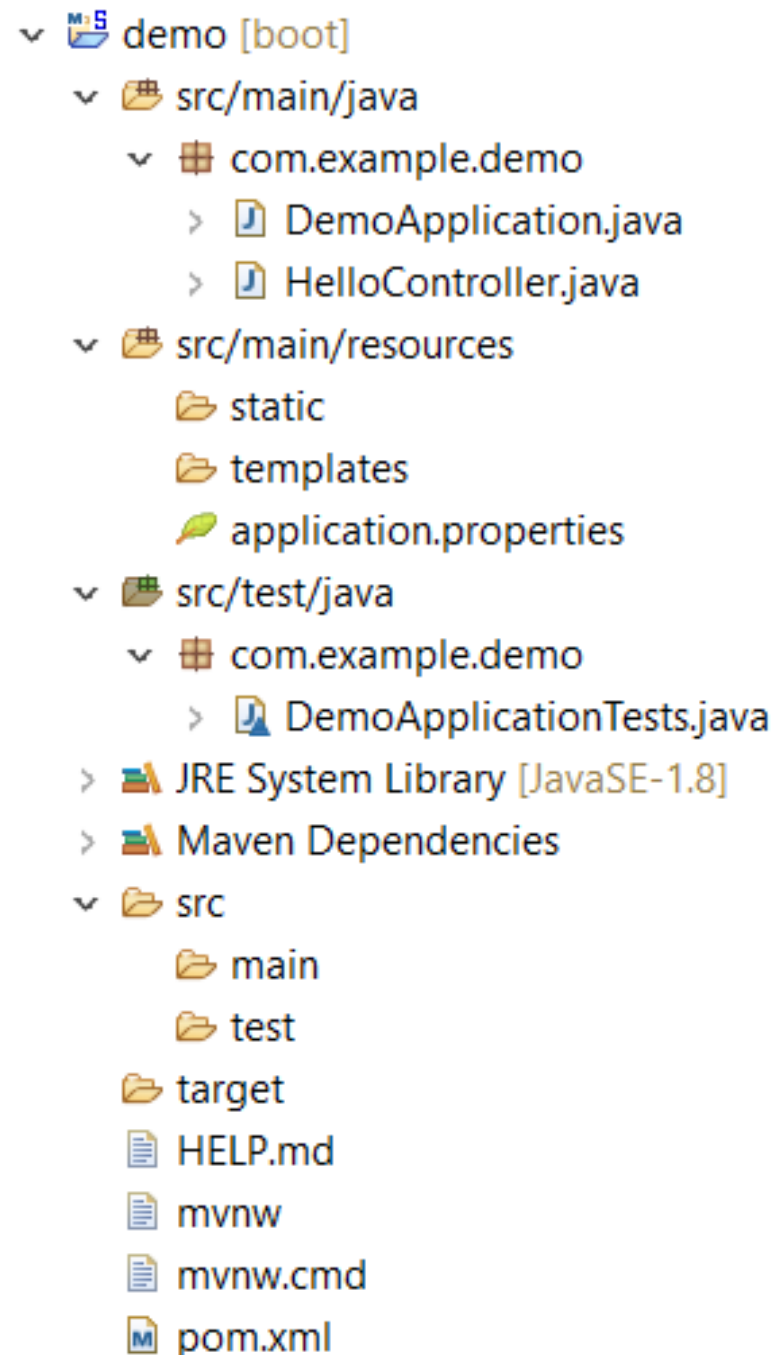


Task 5. SPRING INITIALIZR(Gradle)



Spring Boot Directory Structure

- src/main/java
- src/main/resources
 - static
 - templates
 - application.properties
- src/test/java
- pom.xml



Spring Boot Directory Structure (Cont.)

■ pom.xml

```
<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>
    <exclusions>
      <exclusion>
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-engine</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
</dependencies>
```

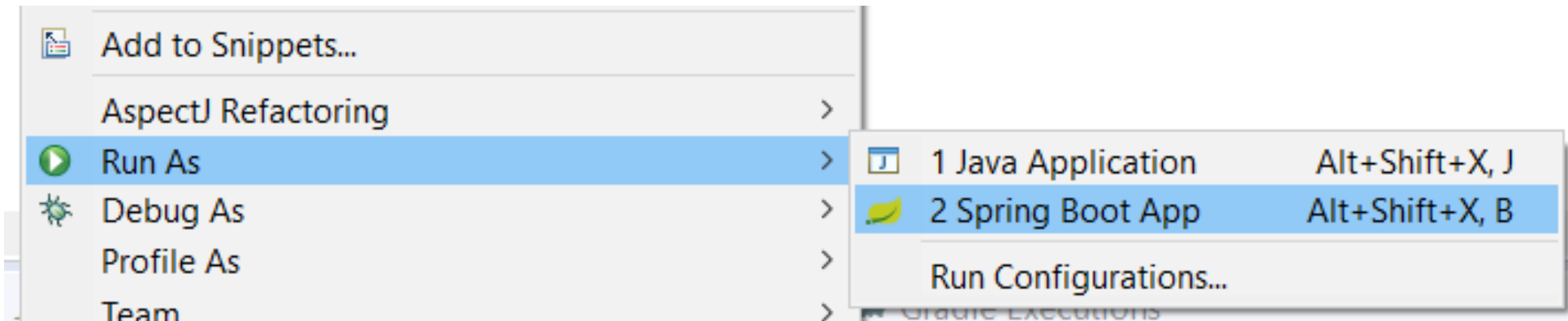
Spring Boot Application Run

- src/main/java/[Project Name + Application.java]

```
1 package com.example.demo;  
2  
3 import org.springframework.boot.SpringApplication;  
4 import org.springframework.boot.autoconfigure.SpringBootApplication;  
5  
6 @SpringBootApplication  
7 public class DemoApplication {  
8  
9     public static void main(String[] args) {  
10         SpringApplication.run(DemoApplication.class, args);  
11     }  
12  
13 }
```

Spring Boot Application Run (Cont.)

- src/main/java/[Project Name + Application.java]
 - J2SE Application 실행 가능
 - J2EE Web Application 실행 가능
 - 내장된 Tomcat 구동
 - **@SpringBootApplication** Annotation
 - 이 Class는 SpringBoot로 만든 Application의 시작 Class임을 의미



Spring Boot Application Run (Cont.)










```
. _ . _ _ _ _  
/\ / _' - _ _ _ _ ( ) _ _ _ _ \\\ \  
( ( )\ _ | '_|'_||'_ V_ | \\ \\ \  
\V _)|_| ||| ||| |( |_| ))))  
' |_||_.|_|_|_|_|_\_,|// ///  
=====|_|=====|___/_=//_/_/_/  
:: Spring Boot ::          (v2.2.1.RELEASE)
```

```

2019-11-07 16:25:10.331 INFO 11580 --- [main] com.example.demo.DemoApplication : Starting DemoApplication on DESKTOP-50VD51T with PID 1158
2019-11-07 16:25:10.331 INFO 11580 --- [main] com.example.demo.DemoApplication : No active profile set, falling back to default profiles: default
2019-11-07 16:25:10.940 INFO 11580 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2019-11-07 16:25:10.946 INFO 11580 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2019-11-07 16:25:10.946 INFO 11580 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.27]
2019-11-07 16:25:10.946 INFO 11580 --- [main] o.a.catalina.core.AprLifecycleListener : Loaded APR based Apache Tomcat Native library [1.2.25] using APR
2019-11-07 16:25:10.946 INFO 11580 --- [main] o.a.catalina.core.AprLifecycleListener : APR capabilities: IPv6 [true], sendfile [true], accept filters [false],
2019-11-07 16:25:10.946 INFO 11580 --- [main] o.a.catalina.core.AprLifecycleListener : APR/OpenSSL configuration: useAprConnector [false], useOpenSSL
2019-11-07 16:25:10.946 INFO 11580 --- [main] o.a.catalina.core.AprLifecycleListener : OpenSSL successfully initialized [OpenSSL 1.1.1c 28 May 2019]
2019-11-07 16:25:10.993 INFO 11580 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2019-11-07 16:25:10.993 INFO 11580 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 631 ms
2019-11-07 16:25:11.118 INFO 11580 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2019-11-07 16:25:11.258 INFO 11580 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2019-11-07 16:25:11.258 INFO 11580 --- [main] com.example.demo.DemoApplication : Started DemoApplication in 1.17 seconds (JVM running for 1.75s)

```

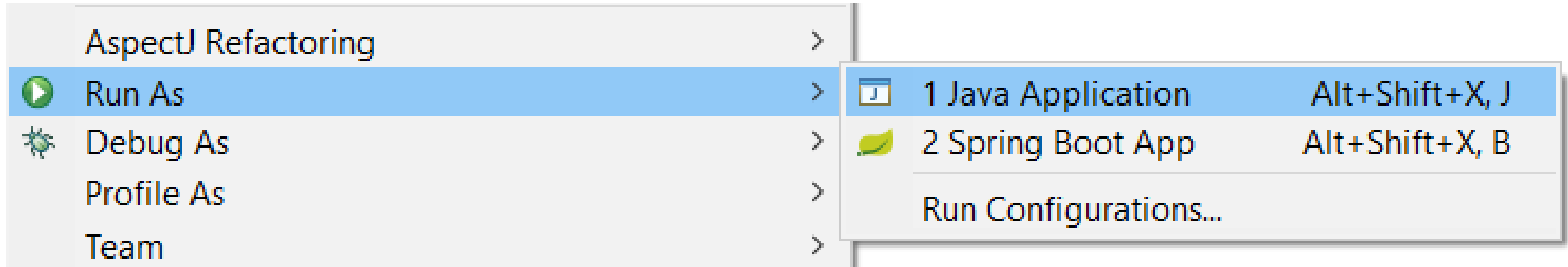
Spring Boot Application Run (Cont.)

- >  jackson-annotations-2.10.0.jar - C:\Users\bluee\.m2\repository\com\fasterxml\jackson\core\jackson-annotations\2.10.0
- >  jackson-core-2.10.0.jar - C:\Users\bluee\.m2\repository\com\fasterxml\jackson\core\jackson-core\2.10.0
- >  jackson-datatype-jdk8-2.10.0.jar - C:\Users\bluee\.m2\repository\com\fasterxml\jackson\datatype\jackson-datatype-jdk8\2.10.0
- >  jackson-datatype-jsr310-2.10.0.jar - C:\Users\bluee\.m2\repository\com\fasterxml\jackson\datatype\jackson-datatype-jsr310\2.10.0
- >  jackson-module-parameter-names-2.10.0.jar - C:\Users\bluee\.m2\repository\com\fasterxml\jackson\module\jackson-module-parameter-names\2.10.0
- >  spring-boot-starter-tomcat-2.2.1.RELEASE.jar - C:\Users\bluee\.m2\repository\org\springframework\boot\spring-boot-starter-tomcat\2.2.1.RELEASE
- >  tomcat-embed-core-9.0.27.jar - C:\Users\bluee\.m2\repository\org\apache\tomcat\embed\tomcat-embed-core\9.0.27
- >  tomcat-embed-el-9.0.27.jar - C:\Users\bluee\.m2\repository\org\apache\tomcat\embed\tomcat-embed-el\9.0.27
- >  tomcat-embed-websocket-9.0.27.jar - C:\Users\bluee\.m2\repository\org\apache\tomcat\embed\tomcat-embed-websocket\9.0.27
- >  spring-boot-starter-validation-2.2.1.RELEASE.jar - C:\Users\bluee\.m2\repository\org\springframework\boot\spring-boot-starter-validation\2.2.1.RELEASE
- >  jakarta.validation-api-2.0.1.jar - C:\Users\bluee\.m2\repository\jakarta\validation\jakarta.validation-api\2.0.1

Spring Boot Application Run (Cont.)

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.WebApplicationType;
5 import org.springframework.boot.autoconfigure.SpringBootApplication;
6
7 @SpringBootApplication
8 public class DemoApplication {
9
10     public static void main(String[] args) {
11         SpringApplication application =
12             new SpringApplication(DemoApplication.class);
13         application.setWebApplicationType(WebApplicationType.NONE);
14         application.run(args);
15     }
16
17 }
```

Spring Boot Application Run (Cont.)



```

      .  _ _ _ _ _
  /\ /  _ _ _ _ _ (.) _ _ _ _ _  \ \ \ \
 ( ( ) \ _ _ | ' _ | ' _ | ' _ V _ | \ \ \ \
  \ V _ _ | _ | | | | | | | ( _ | ) ) ) )
    ' | _ _ | . _ | | | _ | _ \ _ | / / / /
=====|_|=====|_|_/=//_/_/_/
:: Spring Boot ::      (v2.2.1.RELEASE)

```

```
2019-11-07 16:34:12.303 INFO 14384 --- [main] com.example.demo.DemoApplication : Starting DemoApplication on DESKTOP-50VD51T with PID 1438
2019-11-07 16:34:12.303 INFO 14384 --- [main] com.example.demo.DemoApplication : No active profile set, falling back to default profiles: default
2019-11-07 16:34:12.912 INFO 14384 --- [main] com.example.demo.DemoApplication : Started DemoApplication in 0.941 seconds (JVM running for 1.8
```

Spring Boot Application Run (Cont.)

■ `org.springframework.boot.WebApplicationType`

- *NONE*
- *REACTIVE*
- *SERVLET*

```
public static void main(String[] args) {  
    SpringApplication application =  
        new SpringApplication(DemoApplication.class);  
    application.setWebApplicationType(WebApplicationType.);  
}
```

The application should not run as a web application and should not start an embedded web server.

```
enum WebApplicationType {  
    NONE, REACTIVE, SERVLET;  
      
    public static WebApplicationType valueOf(String name) {  
        return valueOf(WebApplicationType.class, name);  
    }  
}
```

Spring Boot Application Run (Cont.)

- src/main/resources/**application.properties**

```
1 ## WebApplication Type Setting
2
3 spring.main.web-application-type=none
```

- **main()** 보다 외부 Properties File의 설정이 우선순위가 더 높다.

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class DemoApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(DemoApplication.class, args);
11     }
12
13 }
```

Spring Boot Application Run (Cont.)

■ Application Banner 변경하기

```

      .
     /\ /  _  '  _  _  (  )  _  _  _  \  \  \  \
    (  ) \  _  |  '  |  '  |  '  |  '  V  _  '  \  \  \  \
     \V /  _  )  |  |  )  |  |  |  |  |  |  (  |  |  )  )  )
      '  |  _  |  .  _  |  |  |  |  |  |  |  \  ,  |  /  /  /  /
=====|_|=====|_|_/=//_/_/_/_/
:: Spring Boot ::      (v2.2.0.RELEASE)

```

```

      .
     /\ /  _  '  _  _  (  )  _  _  _  \  \  \  \
    (  ) \  _  |  '  |  '  |  '  |  '  |  '  V  _  '  \  \  \  \
     \V /  _  )  |  |  )  |  |  |  |  |  |  |  (  |  |  )  )  )
      '  |  _  |  .  _  |  |  |  |  |  |  |  |  \  ,  |  /  /  /  /
=====|_|=====|_|_/=//_/_/_/_/
:: Spring Boot ::      (v2.2.1.RELEASE)

```


Spring Boot Application Run (Cont.)

■ Application Banner 변경하기

```
1 package com.example.demo;
2
3 import org.springframework.boot.Banner;
4 import org.springframework.boot.SpringApplication;
5 import org.springframework.boot.WebApplicationType;
6 import org.springframework.boot.autoconfigure.SpringBootApplication;
7
8 @SpringBootApplication
9 public class DemoApplication {
10
11     public static void main(String[] args) {
12         SpringApplication application = new SpringApplication(DemoApplication.class);
13         application.setWebApplicationType(WebApplicationType.SERVLET);
14         application.setBannerMode(Banner.Mode.OFF);
15         application.run(args);
16     }
17
18 }
```


Spring Boot Application Run (Cont.)

■ 사용자 정의 Banner 사용하기

- src/main/resources/**banner.txt**

```
1 #####
2
3  ※★☐ This is a My Spring Banner ♠☎☹
4
5 #####|
```

- src/main/resources/**application.properties**

```
1 ## WebApplication Type Setting
2 spring.main.web-application-type=servlet
3
4 ## Banner Setting
5 spring.main.banner-mode=console
6
```

Spring Boot Application Run (Cont.)

■ 사용자 정의 Banner 사용하기

```
#####
```

```
※★☐ This is a My Spring Banner ♠☎☪
```

```
#####
```

```
2019-11-07 19:04:21.528 INFO 12464 --- [main] com.example.demo.DemoApplication : Starting
2019-11-07 19:04:21.528 INFO 12464 --- [main] com.example.demo.DemoApplication : No act
2019-11-07 19:04:22.122 INFO 12464 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tom
2019-11-07 19:04:22.134 INFO 12464 --- [main] o.apache.catalina.core.StandardService : Starting
2019-11-07 19:04:22.134 INFO 12464 --- [main] org.apache.catalina.core.StandardEngine : Startin
2019-11-07 19:04:22.134 INFO 12464 --- [main] o.a.catalina.core.AprLifecycleListener : Loaded AP
2019-11-07 19:04:22.134 INFO 12464 --- [main] o.a.catalina.core.AprLifecycleListener : APR capab
```

Spring Boot Application Run (Cont.)

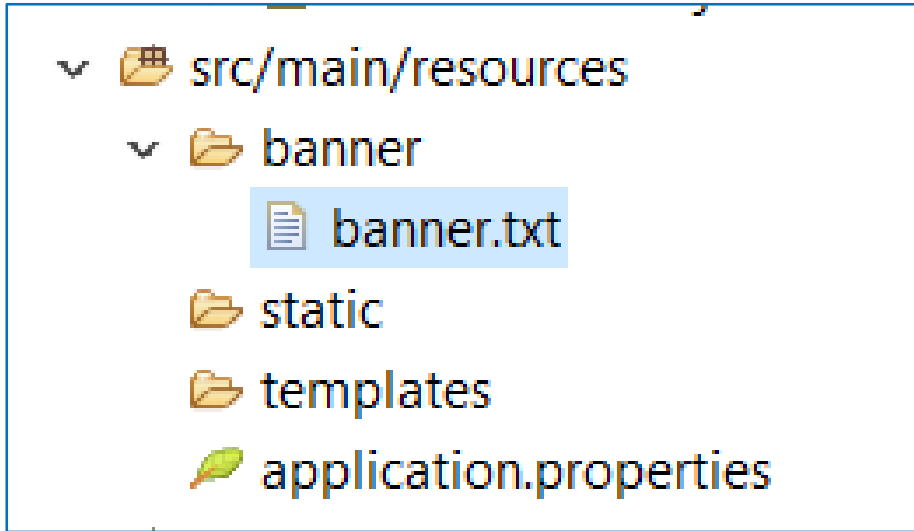
■ 사용자 정의 Banner 사용하기

```
1 #####
2
3  *★☐ This is a My Spring Banner ♠☎☪
4  ${spring-boot.formatted-version}
5
6 #####
7
```

```
#####
*★☐ This is a My Spring Banner ♠☎☪
(v2.2.1.RELEASE)
#####
2019-11-07 19:07:02.504 INFO 2500 --- [main] com.example.demo.DemoAppl
2019-11-07 19:07:02.520 INFO 2500 --- [main] com.example.demo.DemoAppl
2019-11-07 19:07:03.098 INFO 2500 --- [main] o.s.b.w.embedded.tomcat.Tom
2019-11-07 19:07:03.102 INFO 2500 --- [main] o.apache.catalina.core.Standa
```

Spring Boot Application Run (Cont.)

■ banner.txt 위치 변경하기



- **application.properties** 수정

```
1 ## WebApplication Type Setting
2 spring.main.web-application-type=servlet
3
4 ## Banner Setting
5 spring.main.banner-mode=console
6 spring.banner.location=banner/banner.txt
7
```

Spring Boot Application Run (Cont.)

■ Image Banner 적용하기

- src/main/resources
 - banner
 - banner.txt
 - Spring.jpg
 - static
 - templates
 - application.properties



● application.properties 수정

```
1 ## WebApplication Type Setting
2 spring.main.web-application-type=servlet
3
4 ## Banner Setting
5 spring.main.banner-mode=console
6 spring.banner.location=banner/banner.txt
7 spring.banner.image.location=banner/Spring.jpg
8 spring.banner.image.width=80
9 spring.banner.image.height=10
```

Spring Boot Application Run (Cont.)

■ Image Banner 적용하기

```

  o.@@o&
  :  o:*
  :*
    @&&&#@          .*8:8:~*
    8::  @##  &*  ::  #o@  @8@*  @&  :@##  ##@*  @*  @:
  ...  *8  @o&  @*:  *:  @oo  :8  @&o@  &&@  :#  .8
8  **  &..  @:&  &&  &&  &.8  :&  @&o@  @  &o@  .&
&      8...@:8  oo:  &  8  *.  @#8  @  88#  *..  *@.*
*      ...88&  .**  o.  #@@8*  ###  @  #&&  ..  **
#o      *..o  @####  &oo&&  @ooo8  @  @8#8  8888o  #:&&&#@  o.*
  @***&#                                     ..#

#####

※★☐ This is a My Spring Banner ♠☎☪
(v2.2.1.RELEASE)

#####
2019-11-07 19:18:13.228 INFO 6488 --- [          main] com.example.demo.DemoApplication
```

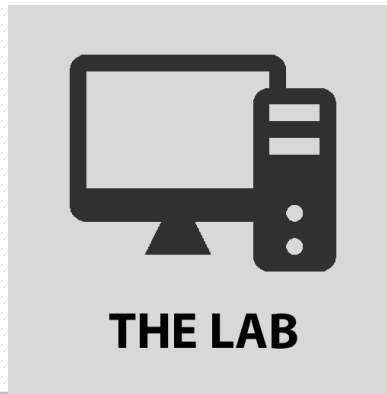
Spring Boot Application Run (Cont.)

■ Tomcat Server Port 변경하기

- **application.properties**

```
1 ## WebApplication Type Setting
2 spring.main.web-application-type=servlet
3
4 ## Tomcat Server Setting
5 server.port=8000
```

```
] com.example.demo.DemoApplication : No active profile set, falling back to default pr
] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8000 (http)
] o.apache.catalina.core.StandardService : Starting service [Tomcat]
] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.2
] o.a.catalina.core.AprLifecycleListener : Loaded APR based Apache Tomcat Native library [
] o.a.catalina.core.AprLifecycleListener : APR capabilities: IPv6 [true], sendfile [true], acce
] o.a.catalina.core.AprLifecycleListener : APR/OpenSSL configuration: useAprConnector [fa
] o.a.catalina.core.AprLifecycleListener : OpenSSL successfully initialized [OpenSSL 1.1.1c
] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContex
] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization comple
] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskEx
] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8000 (http) with
] com.example.demo.DemoApplication : Started DemoApplication in 1.144 seconds (1)
```



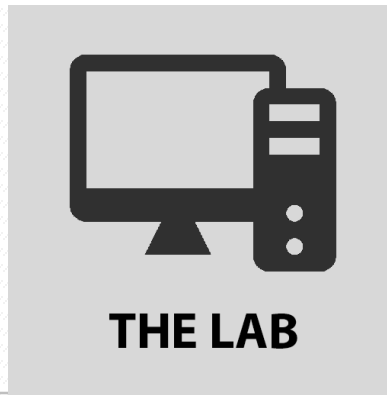
Task 6. 사용자 정의 Starter 만들기





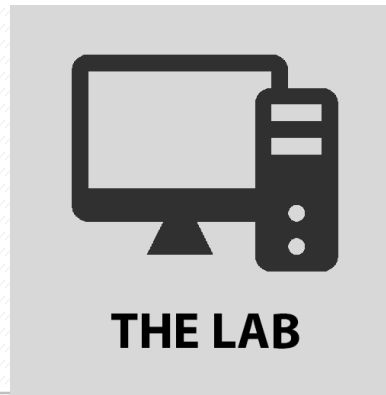
Task 7. 간단한 JPA Project





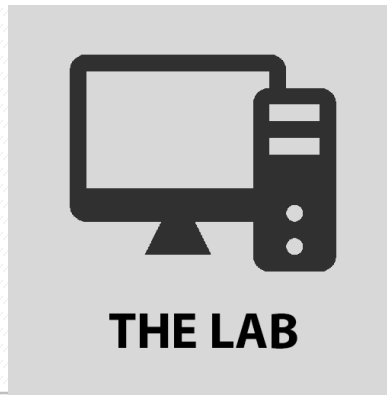
Task 8. 정적 Page 만들기





Task9. JSP Page 만들기





Task10. thymeleaf template 사용하기

Templates

■ Thymeleaf

- **th:00** 형식으로 속성을 HTML tag에 추가해서 값이나 처리 등을 page에 심을 수 있다.
- 자체 tag를 사용하지 않으므로 HTML visual editor와도 궁합이 좋으며 편집이 쉽다는 장점이 있다.
- Spring Boot에서는 이 template을 선택하는 사람이 가장 많다고 한다.
- 기본이 되는 HTML 위에 필요한 속성을 추가만 하면 되기 때문에 간단하고 이해하기 쉬운 구성으로 되어 있다.
- 따라서 template 관련 새로운 기술을 배우지 않고서도 바로 사용할 수 있어 학습에 대한 부담감이 줄어든다.

Templates (Cont.)

■ JSP

- Spring Boot에서는 기본적으로 사용하지 않는다.
- Spring Boot에서는 application을 Java Server와 함께 Jar file로 묶어 배포하는 경우가 많지만 이 방법에서는 JSP가 동작하지 않는다.
- 옛날 방식인, 이미 Java Server가 실행되고 있고 그 위에 War file을 upload하는 경우에만 동작한다.
- 최근 cloud를 사용해서 app을 배포하는 경우가 늘고 있는데, 이 경우에는 jar file을 이용하기 때문에 JSP를 사용할 수 없는 경우가 늘고 있다.

Templates (Cont.)

■ FreeMarker

- 이것은 **`${}`** 형식으로 값을 채워나가는 template이다.
- Thymeleaf가 특수한 속성으로 값을 선정하는 것에 비해 이 template은 text를 표시할 위치에 직접 **`${}`**를 넣어야 표시할 수 있다.
- 또한 제어를 위해서 HTML tag와 같은 **`<#00>`** 형식으로 tag를 사용하므로 HTML Visual Editor에서는 HTML 구조에 영향을 끼칠 가능성이 있다.

■ Groovy

- 다른 template과 달리 HTML 기반으로 하지 않고 HTML과는 전혀 다른 code 체계를 가지고 있기 때문에 HTML밖에 모르는 개발자의 입장에서는 적합하지 않다.
- code를 이용하여 화면 전체를 작성하는 것이 편한 사람에게 권장.

Templates (Cont.)

■ Velocity

- Apache Software Foundation이 개발 중인 template engine이다.
- **\$**나 **#**같은 특수한 문자를 사용해서 변수 등을 HTML code 내에 직접 작성할 수 있다.
- FreeMarker와 비슷하지만, FreeMarker가 tag 형태로 구문을 작성하는 것에 비해 이것은 tag 없이 바로 작성할 수 있어서 Visual Editor 등에 영향이 적다.

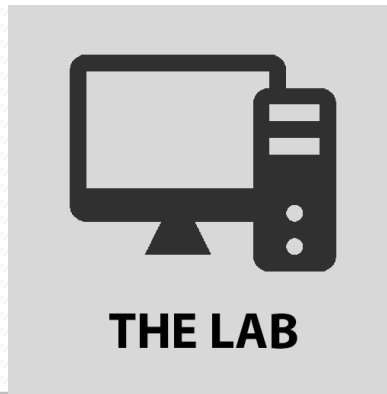
■ Mustache

- 다양한 언어에 적용할 수 있는 template이다.
- Java 뿐만 아니라 PHP, Ruby, Python, JavaScript 등에서 사용할 수 있어서 Java 이외의 언어를 사용하고 싶은 사람에게 좋은 대안이 될 수 있다.
- **{{}}** 기호를 사용해서 변수 등을 HTML code 내에 기술한다.

Templates (Cont.)

■ 기본은 변수식과 OGNL

- Thymeleaf의 기본은 '값을 출력하는(표시하는) 것'이다.
- 이것은 **`${00}`** 형식으로 작성
- 이 형식을 **변수식**이라고 부른다.
- **변수식** 안에 작성하는 것은 **OGNL(Object-Graph Navigation Language) 식**이다.
- **OGNL**은 Java의 값에 접근하기 위한 식이다.
- Thymeleaf뿐만 아니라 다양한 library와 framework 등에서 사용되고 있다.



Task11. thymeleaf template 사용하기2

Utility Object

- **변수식**은 이름 그대로 변수를 기술해서 그대로 출력할 수 있다.
- 이 변수는 이미 예제에서 확인했듯이, Controller에서 값을 준비하고 그것을 Template 측에 전달해서 출력하는 것이 기본적인 흐름이다.
- 하지만, Java class 중에는 이런 Template에서 사용 빈도가 높은 것이 있다.
- 이런 Class를 사용할 때도 항상 Controller에서 Class를 변수로 준비하고...하는 것이 매우 번거로운 일이다.
- 그래서 Thymeleaf에서는 자주 사용되는 class를 **#이름** 이라는 상수로 정의해서 **변수식** 안에 직접 작성할 수 있게 하고 있다.
- 이것을 *Utility Object*라고 한다.

Utility Object (Cont.)

■ 종류

- **#strings** : String class의 상수
- **#numbers** : Number class의 상수
- **#bools** : Boolean class의 상수
- **#dates** : Date class의 상수
- **#objects** : Object class의 상수
- **#arrays** : Array class의 상수
- **#lists** : List class의 상수
- **#sets** : Set class의 상수
- **#maps** : Map class의 상수

Utility Object (Cont.)

- 이 객체들은 Class의 상수이므로 직접 Class Method등을 호출해서 사용할 수 있다.
- 단, **new #dates**처럼 하면 Date Instance를 만들 수 없다.
- **#dates.00**과 같이 Class Field나 Method 호출 등에 사용한다.

매개변수에 접근하기

- 만일 Controller를 거치지 않고 Query String 같은 매개변수를 사용하고자 할 때는 **param** 이라는 변수를 사용한다.
- 이것은 **변수식** 안에서 직접 사용할 수 있는 변수이다.
- 이 변수 안에 있는 매개변수명을 지정해서 추출할 수 있다.
- **`${param.id}`**라고 하면 **`id=javaexpert`**의 형태로 전송된 값을 받을 수 있다.
- 독특하게도 이 방식은 배열 형태로 되어 있어서 배열내에서 추출해야 한다.

Message식

- **`${}`**라는 **변수식** 외에도 thymeleaf에서 사용할 수 있는 Message식이 있다.
- 이것은 Project에 미리 준비해둔 properties File(설정 file)에서 값을 가져와서 표시하는 것이다.
- Java에서는 properties File에 미리 Text(설정값)을 저장해두고, 필요에 따라 File 내의 Text값을 꺼내서 사용하는 경우가 있다.
- 이것을 Template에서 사용할 수 있게 한 것이 Message식이다.
- 형식은 아래와 같다.

`#{값 지정}`

- **`${}`**가 아니라 **`#{}`**로 작성하는 것이 특징이다.

Link식과 href

- Web Page에서는 URL을 지정하는 Link도 다양한 곳에 사용되며 이 Link를 지정하기 위한 전용식도 존재한다.

@{주소}

- 이것은 기본적으로 URL을 지정하는 속성(<a> tag의 href 등)에서 사용한다.
- 예를 들어, @{/index}라고 작성하면 /index로 가는 Link를 지정할 수 있다.

선택 객체와 변수식

- **변수식**에서 객체를 사용하려고 하면 문제가 발생한다.
- 물론 **`${object.name}`**처럼 객체의 property나 method를 지정해서 작성하면 되지만 객체안에서 다수의 값이 존재하는 경우 일일이 작성하는 것은 매우 귀찮은 일이다.
- 이런 경우 객체를 지정해서 해당 객체 안에 있는 값을 추출할 수 있는 전용 **변수식**을 사용하면 된다.

`*{}`

선택 객체와 변수식 (Cont.)

- 이 변수식은 객체를 처리하는 **변수식** 내부에서 사용한다.

<00 th:object="\${객체}">

<XX th:text="*{property}">

</00>

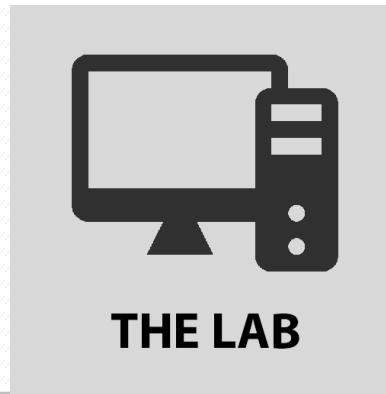
- tag에 **th:object**라는 속성을 사용해서 객체를 설정한다.
- 이렇게 하면 해당 tag 내부에서 ***{}**를 이용한 **변수식**을 사용할 수 있게 된다.
- 이 ***** 형태의 **변수식**에선 객체 내의 Property 등을 이름만으로도 지정할 수 있다.



Task12. Spring Boot와 JDBC 연동하기



Task13. Spring Boot와 JPA 연동하기



Task14. Spring Boot JPA

