

```
1 HOL : Spring Boot
2 -----
3 Task1. Maven으로 Spring Boot Project 생성하기
4 1. In Project Explorer
5 1)right-click > New > Project > Maven > Maven Project > Next
6 2)[New Maven project] 창에서 > Next
7 -Group Id : org.apache.maven.archetypes
8 -Artifact Id : maven-archetype-quickstart
9 -Version : 1.1
10 -Next
11 3)Group Id : com.example
12 -Artifact Id : springbootdemo
13 -Version : 0.0.1-SNAPSHOT
14 -Package : com.example.springbootdemo
15 -Finish
16
17 2. pom.xml 수정
18 <project xmlns="http://maven.apache.org/POM/4.0.0"
19 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
20 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
21 <modelVersion>4.0.0</modelVersion>
22
23 <groupId>com.example</groupId>
24 <artifactId>springbootdemo</artifactId>
25 <version>0.0.1-SNAPSHOT</version>
26 <packaging>jar</packaging>
27
28 <name>springbootdemo</name>
29 <url>http://maven.apache.org</url>
30 <parent>
31 <groupId>org.springframework.boot</groupId>
32 <artifactId>spring-boot-starter-parent</artifactId>
33 <version>2.2.0.RELEASE</version>
34 </parent>
35
36 <properties>
37 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
38 <java.version>1.8</java.version>
39 </properties>
40
41 <dependencies>
42 <dependency>
43 <groupId>junit</groupId>
44 <artifactId>junit</artifactId>
45 <version>4.12</version>
46 <scope>test</scope>
47 </dependency>
48 <dependency>
49 <groupId>org.springframework.boot</groupId>
50 <artifactId>spring-boot-starter-web</artifactId>
51 </dependency>
```

```

52     <dependency>
53         <groupId>org.springframework.boot</groupId>
54         <artifactId>spring-boot-starter-test</artifactId>
55         <scope>test</scope>
56     </dependency>
57 </dependencies>
58 </project>
59
60 1)<parent> 설정하기
61     -Spring Boot의 설정 정보를 상속한다.
62     -여기서 지정한 version이 spring boot의 version이 된다.
63     -spring boot의 version을 올리려면 <version> tag 안에 있는 설정 값을 변경한다.
64
65 2)spring-boot-starter-web
66     -spring boot로 web application을 만들 때 참조할 기본 library 정보를 설정한다.
67     -이렇게 쓰기만 해도 web application 제작에 필요한 spring framework 관련 library와 third-party
68     library를 이용할 수 있게 된다.
69     -version은 위 parent에서 설정한 spring-boot-starter-parent 안에 정의되어 있으므로, 여기서는 지정하
70     지 않아도 된다.
71
72 3)pop.xml > right-click > Run As > Maven install
73
74 3. Project > right-click > Properties > Project Facets 수정하기
75 1)Java --> 1.8
76 2)Runtimes Tab
77     -Apache Tomcat v9.0 check
78     -jdk1.8.0_221 check
79 3)Apply and Close
80
81 4. Project > right-click > Maven > Update Project... > OK
82
83 5. Hello World!를 출력하는 Web application 작성하기
84 1)src/main/java/com/example/springbootdemo/App.java
85
86     package com.example.springbootdemo;
87
88     import org.springframework.boot.SpringApplication;
89     import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
90     import org.springframework.web.bind.annotation.RequestMapping;
91     import org.springframework.web.bind.annotation.RestController;
92
93     /**
94      * Hello world!
95      *
96      */
97     @RestController
98     @EnableAutoConfiguration
99     public class App {
100         @RequestMapping("/")
101         String home(){

```

```

102         return "Hello, World!";
103     }
104
105     public static void main( String[] args ){
106         SpringApplication.run(App.class, args);
107     }
108 }
109
110 2)@RestController
111     -이 annotation을 붙이면 web application에서 request를 받아들이는 controller class임을 나타낸다.
112
113 3)@EnableAutoConfiguration
114     -이 annotation은 매우 중요하다.
115     -이 annotation을 붙이면 다양한 설정이 자동으로 수행되고 기존의 spring application에 필요했던 설정
116     file들이 필요없게 된다.
117
118 4)@RequestMapping("/")
119     -이 annotation이 붙으면 이 method가 HTTP 요청을 받아들이는 method임을 나타낸다.
120     -@GetMapping도 가능
121     -@RequestMapping(value="/", method=RequestMethod.GET)과 @GetMapping은 동일하다.
122
123 5)return "Hello World!";
124     -HTTP 응답을 반환한다.
125     -@RestController annotation이 붙은 class에 속한 method에서 문자열을 반환하면 해당 문자열이 그대로
126     HTTP 응답이 되어 출력된다.
127
128 6)SpringApplication.run(App.class, args);
129     -spring boot application을 실행하는 데 필요한 처리를 main() 안에서 작성한다.
130     -@EnableAutoConfiguration annotation이 붙은 class를 SpringApplication.run()의 첫번째 인자로 지정
131     한다.
132
133 6. Web Application 실행하기
134
135 1)springbootdemo project > right-click > Run As > Spring Boot App
136
137
138
139
140
141
142
143
144
145

```

```

. ____ - _ _ _
/WW / _' _ _ _(_)_ _ _ WW WW WW
( )W_ | ' | ' | ' W/ _ | WW WW WW
WW/ _)| | | | | | ( | ) ) )
' | _ | _ | | | | _W_ | / / / /
=====|_|=====|_|=/ / / /
:: Spring Boot ::      (v2.2.0.RELEASE)

2019-11-06 19:51:10.778 INFO 1332 --- [           main] com.example.springbootdemo.App :
Starting App on DESKTOP-50VD51T with PID 1332
(C:\SpringHome\springbootdemo\target\classes started by instructor in
C:\SpringHome\springbootdemo)
2019-11-06 19:51:10.794 INFO 1332 --- [           main] com.example.springbootdemo.App :
No active profile set, falling back to default profiles: default
2019-11-06 19:51:12.965 INFO 1332 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer
: Tomcat initialized with port(s): 8080 (http)
2019-11-06 19:51:12.997 INFO 1332 --- [           main] o.apache.catalina.core.StandardService :

```

```

Starting service [Tomcat]
146 2019-11-06 19:51:12.997 INFO 1332 --- [      main] org.apache.catalina.core.StandardEngine :
Starting Servlet engine: [Apache Tomcat/9.0.27]
147 2019-11-06 19:51:12.997 INFO 1332 --- [      main] o.a.catalina.core.AprLifecycleListener :
Loaded APR based Apache Tomcat Native library [1.2.23] using APR version [1.7.0].
148 2019-11-06 19:51:12.997 INFO 1332 --- [      main] o.a.catalina.core.AprLifecycleListener : APR
capabilities: IPv6 [true], sendfile [true], accept filters [false], random [true].
149 2019-11-06 19:51:12.997 INFO 1332 --- [      main] o.a.catalina.core.AprLifecycleListener :
APR/OpenSSL configuration: useAprConnector [false], useOpenSSL [true]
150 2019-11-06 19:51:13.013 INFO 1332 --- [      main] o.a.catalina.core.AprLifecycleListener :
OpenSSL successfully initialized [OpenSSL 1.1.1c 28 May 2019]
151 2019-11-06 19:51:13.216 INFO 1332 --- [      main] o.a.c.c.C.[Tomcat].[localhost].[/] :
Initializing Spring embedded WebApplicationContext
152 2019-11-06 19:51:13.216 INFO 1332 --- [      main] o.s.web.context.ContextLoader :
Root WebApplicationContext: initialization completed in 2282 ms
153 2019-11-06 19:51:13.653 INFO 1332 --- [      main] o.s.s.concurrent.ThreadPoolTaskExecutor :
Initializing ExecutorService 'applicationTaskExecutor'
154 2019-11-06 19:51:14.091 INFO 1332 --- [      main] o.s.b.w.embedded.tomcat.TomcatWebServer
: Tomcat started on port(s): 8080 (http) with context path ""
155 2019-11-06 19:51:14.105 INFO 1332 --- [      main] com.example.springbootdemo.App :
Started App in 4.263 seconds (JVM running for 6.681)

```

```

156
157 2)출력된 log 내용을 보면 8080 port로 tomcat이 시작된다는 것을 알 수 있다.
158 3)SpringApplication.run() method에서 내장 server를 시작했기 때문이다.
159 4)http://localhost:8080/로 접속해보자.
160 5)Web browser에 'Hello, World!'가 출력된다.
161 6)application을 끝내려면 Ctrl + C를 누르고, '[일괄 작업을 끝내시겠습니까 (Y/N)?'라는 질문에 'y'를 입력
하고 enter key를 누르면 된다.
162 7)여기서 알게 된 사실
163   -설정할 의존 관계의 갯수가 적다.
164   -Java Class 하나만 작성하면 된다.
165   -명령 prompt에서 application을 실행한다.
166
167
168 -----

```

169 Task2. STS로 Spring Boot Application 개발하기

170 1. Package Explorer > right-click > New > Spring Starter Project

```

171 1)Service URL :http://start.spring.io
172 2)Name : demo
173 3)Type : Maven
174 4)Packaging : jar
175 5)Java Version : 8
176 6)Language : Java
177 7)Group : com.example
178 8)Artifact : demo
179 9)Version : 0.0.1-SNAPSHOT
180 10)Description : Demo project for Spring Boot
181 11)Package : com.example.demo
182
183 12)Next
184

```

185 2. [New Spring Starter Project Dependencies] 창에서

```

186 1)Spring Boot Version : 2.2.0
187 2)Select Web > check Spring Web > Finish
188
189 3. src/main/java/com.example.demo.DemoApplication.java
190
191 package com.example.demo;
192
193 import org.springframework.boot.SpringApplication;
194 import org.springframework.boot.autoconfigure.SpringBootApplication;
195
196 @SpringBootApplication
197 public class DemoApplication {
198
199     public static void main(String[] args) {
200         SpringApplication.run(DemoApplication.class, args);
201     }
202
203 }
204
205 4. DemoApplication.java > right-click > Run As > Spring Boot App
206
207 . ____ - _ _ _
208 /WW / _' _ _ _(_) _ _ _ WW WW WW
209 (( )W _ | ' | ' | ' _ W/ _ | WW WW WW
210 WW/ _ )| | )| | | | | (| | ) ) )
211 ' | _ | _ | | | | _ W _ | / / / /
212 =====|_|=====|_|=/ / / / /
213 :: Spring Boot :: (v2.2.0.RELEASE)
214
215 2019-11-06 20:01:51.977 INFO 11800 --- [ main] com.example.demo.DemoApplication :
Starting DemoApplication on DESKTOP-50VD51T with PID 11800
(C:\WSpringHome\demo\target\classes started by instructor in C:\WSpringHome\demo)
216 2019-11-06 20:01:51.977 INFO 11800 --- [ main] com.example.demo.DemoApplication :
No active profile set, falling back to default profiles: default
217 2019-11-06 20:01:54.305 INFO 11800 --- [ main] o.s.b.w.embedded.tomcat.TomcatWebServer :
Tomcat initialized with port(s): 8080 (http)
218 2019-11-06 20:01:54.332 INFO 11800 --- [ main] o.apache.catalina.core.StandardService :
Starting service [Tomcat]
219 2019-11-06 20:01:54.332 INFO 11800 --- [ main] org.apache.catalina.core.StandardEngine :
Starting Servlet engine: [Apache Tomcat/9.0.27]
220 2019-11-06 20:01:54.332 INFO 11800 --- [ main] o.a.catalina.core.AprLifecycleListener :
Loaded APR based Apache Tomcat Native library [1.2.23] using APR version [1.7.0].
221 2019-11-06 20:01:54.332 INFO 11800 --- [ main] o.a.catalina.core.AprLifecycleListener : APR
capabilities: IPv6 [true], sendfile [true], accept filters [false], random [true].
222 2019-11-06 20:01:54.332 INFO 11800 --- [ main] o.a.catalina.core.AprLifecycleListener :
APR/OpenSSL configuration: useAprConnector [false], useOpenSSL [true]
223 2019-11-06 20:01:54.347 INFO 11800 --- [ main] o.a.catalina.core.AprLifecycleListener :
OpenSSL successfully initialized [OpenSSL 1.1.1c 28 May 2019]
224 2019-11-06 20:01:54.535 INFO 11800 --- [ main] o.a.c.c.C.[Tomcat].[localhost].[/] :
Initializing Spring embedded WebApplicationContext
225 2019-11-06 20:01:54.535 INFO 11800 --- [ main] o.s.web.context.ContextLoader : Root
WebApplicationContext: initialization completed in 2402 ms

```

```

226 2019-11-06 20:01:54.957 INFO 11800 --- [      main] o.s.s.concurrent.ThreadPoolTaskExecutor :
      Initializing ExecutorService 'applicationTaskExecutor'
227 2019-11-06 20:01:55.410 INFO 11800 --- [      main] o.s.b.w.embedded.tomcat.TomcatWebServer :
      Tomcat started on port(s): 8080 (http) with context path ""
228 2019-11-06 20:01:55.418 INFO 11800 --- [      main] com.example.demo.DemoApplication :
      Started DemoApplication in 4.362 seconds (JVM running for 6.603)
229
230
231 5. http://localhost:8080
232 Whitelabel Error Page
233 This application has no explicit mapping for /error, so you are seeing this as a fallback.
234
235 Wed Nov 06 20:02:04 KST 2019
236 There was an unexpected error (type=Not Found, status=404).
237 No message available
238
239
240 6. src/main/java/com.example.demo.DemoApplication.java 수정하기
241
242 package com.example.demo;
243
244 import org.springframework.boot.SpringApplication;
245 import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
246 import org.springframework.web.bind.annotation.RequestMapping;
247 import org.springframework.web.bind.annotation.RestController;
248
249 @RestController
250 @EnableAutoConfiguration
251 public class DemoApplication {
252
253     @RequestMapping("/")
254     String home() {
255         return "Hello, World!";
256     }
257
258     public static void main(String[] args) {
259         SpringApplication.run(DemoApplication.class, args);
260     }
261 }
262
263 1) DemoApplication.java > right-click > Run As > Spring Boot App
264 2) http://localhost:8080/
265
266 Hello, World!
267
268
269 -----
270 Task3. Groovy로 Application 개발하기
271 1. 준비
272 1) Visit
      https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html

```

Page 7 of 76

```

: Unknown class loader
[org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentClassLoader@
5815d247] of class [class
org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentClassLoader]
316 2019-11-07 14:47:19.254 INFO 13524 --- [runner-0] o.a.c.c.C.[Tomcat].[localhost].[/] :
    Initializing Spring embedded WebApplicationContext
317 2019-11-07 14:47:19.254 INFO 13524 --- [runner-0] o.s.web.context.ContextLoader :
    Root WebApplicationContext: initialization completed in 1110 ms
318 2019-11-07 14:47:19.441 INFO 13524 --- [runner-0] o.s.s.concurrent.ThreadPoolTaskExecutor :
    Initializing ExecutorService 'applicationTaskExecutor'
319 2019-11-07 14:47:19.837 INFO 13524 --- [runner-0]
    o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context
    path "
320 2019-11-07 14:47:19.837 INFO 13524 --- [runner-0] o.s.boot.SpringApplication :
    Started application in 2.405 seconds (JVM running for 4.416)
321
322
323 3. Web browser로 http://localhost:8080에 접속한다.
324
325 Hello!!!
326
327
328 4. app.groovy code 수정하기
329 1)Command Prompt 에서 Ctrl + C를 눌러서 종료시킨다.
330 2)일괄 작업을 끝내시겠습니까 (Y/N)? Y
331 3)아래와 같이 code를 수정한다.
332
333 @RestController
334 class App {
335
336     @RequestMapping("/")
337     def home() {
338         def header = "<html><body>"
339         def footer = "</body></html>"
340         def content = "<h1>Hello! Spring Boot with Groovy</h1><p>This is html content.</p>"
341
342         header + content + footer
343     }
344 }
345
346
347 5. 다시 script를 실행한다.
348
349 $ spring run app.groovy
350
351 6. Browser를 refresh 한다.
352
353 Hello! Spring Boot with Groovy
354 This is html content.
355
356 7. Template 사용하기
357 1)template은 HTML을 기반으로 작성된 code를 읽어 rendering해서 web page에 출력하는 기능이다.

```


358 2)이런 기능의 template이 몇 가지 종류가 있지만, spring boot에서는 thymeleaf(타임리프)라고 하는 library를 자주 사용한다.

359 3)<http://www.thymeleaf.org>

360 4)template file 작성

361 5)C:\temp\templates\home.html

```
362
363 <!doctype html>
364 <html lang="en">
365   <head>
366     <meta charset="UTF-8" /> <!--반드시 종결 tag 필요 -->
367     <title>Index Page</title>
368     <style type="text/css">
369       h1 { font-size:18pt; font-weight:bold; color:gray; }
370       body { font-size:13pt; color:gray; margin:5px 25px; }
371     </style>
372   </head>
373   <body>
374     <h1>Hello! Spring Boot with Thymeleaf</h1>
375     <p>This is sample web page.</p>
376   </body>
377 </html>
```

378

379 6)template file은 controller가 있는 곳의 templates folder 안에 두어야 한다.

380 7)controller 수정하기

381

382 -app.groovy

383

384 @Grab("thymeleaf-spring5")

385

386 @Controller

387 class App {

388

389 @RequestMapping("/")

390 @ResponseBody

391 def home(ModelAndView mav) {

392 mav.setViewName("home")

393 mav

394 }

395 }

396

397 8)다시 script 실행

398

399 \$ spring run app.groovy

400

401 9)<http://localhost:8080>

402

403 Hello! Spring Boot with Thymeleaf

404 This is sample web page.

405

406

407 8. form 전송하기

408 1)home.html

```
409
410 <!doctype html>
411 <html lang="en">
412   <head>
413     <meta charset="UTF-8" />
414     <title>Index Page</title>
415     <style type="text/css">
416       h1 { font-size:18pt; font-weight:bold; color:gray; }
417       body { font-size:13pt; color:gray; margin:5px 25px; }
418     </style>
419   </head>
420   <body>
421     <h1>Hello!</h1>
422     <p th:text="{msg}">${msg}</p>
423     <form method="post" action="/send">
424       <input type="text" name="text1" th:value="{value}" />
425       <input type="submit" value="Send" />
426     </form>
427   </body>
428 </html>
429
430 2)app.groovy
431
432 @Grab("thymeleaf-spring5")
433
434 @Controller
435 class App {
436
437   @RequestMapping(value = "/", method=RequestMethod.GET)
438   @ResponseBody
439   def home(ModelAndView mav) {
440     mav.setViewName("home")
441     mav.addObject("msg", "Please write your name...")
442     mav
443   }
444
445   @RequestMapping(value = "/send", method=RequestMethod.POST)
446   @ResponseBody
447   def send(@RequestParam("text1") String str, ModelAndView mav){
448     mav.setViewName("home")
449     mav.addObject("msg", "Hello, " + str + "!!!")
450     mav.addObject("value", str)
451     mav
452   }
453 }
454
455 3)script 실행
456 $ spring run app.groovy
457
458 4)<a href="http://localhost:8080">http://localhost:8080
459
460 Hello!
```

```
461     Hello, 한지민!!!
462
463
464 -----
465 Task4. SPRING INITIALIZR(Maven)
466 1. Visit http://start.spring.io/
467 2. 설정
468     1)Maven Project
469     2)Java
470     3)2.2.1
471     4)Group : com.example
472     5)Artifact : demo
473     6)Name : demo
474     7)Description : Demo project for Spring Boot
475     8)Package Name : com.example.demo
476     9)Packaging : Jar
477     10)Java Version : 8
478     11)Dependencies : Web
479
480     12)Click [Generate]
481     13)Downloads [demo.zip] : 55.5KB
482     14)Unpack to Spring workspace.
483
484
485 3. Project Import
486     1)In Package Explorer > right-click > Import > Maven > Existing Maven Projects > Next
487     2)Click [Browse...] > demo Folder Select > Finish
488     3)pom.xml
489
490     <?xml version="1.0" encoding="UTF-8"?>
491     <project xmlns="http://maven.apache.org/POM/4.0.0"
492     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
493     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
494     https://maven.apache.org/xsd/maven-4.0.0.xsd">
495     <modelVersion>4.0.0</modelVersion>
496     <parent>
497     <groupId>org.springframework.boot</groupId>
498     <artifactId>spring-boot-starter-parent</artifactId>
499     <version>2.2.1.RELEASE</version>
500     <relativePath/> <!-- lookup parent from repository -->
501     </parent>
502     <groupId>com.example</groupId>
503     <artifactId>demo</artifactId>
504     <version>0.0.1-SNAPSHOT</version>
505     <name>demo</name>
506     <description>Demo project for Spring Boot</description>
507
508     <properties>
509     <java.version>1.8</java.version>
510     </properties>
511
512     <dependencies>
```

```

511     <dependency>
512         <groupId>org.springframework.boot</groupId>
513         <artifactId>spring-boot-starter-web</artifactId>
514     </dependency>
515
516     <dependency>
517         <groupId>org.springframework.boot</groupId>
518         <artifactId>spring-boot-starter-test</artifactId>
519         <scope>test</scope>
520         <exclusions>
521             <exclusion>
522                 <groupId>org.junit.vintage</groupId>
523                 <artifactId>junit-vintage-engine</artifactId>
524             </exclusion>
525         </exclusions>
526     </dependency>
527 </dependencies>
528
529 <build>
530     <plugins>
531         <plugin>
532             <groupId>org.springframework.boot</groupId>
533             <artifactId>spring-boot-maven-plugin</artifactId>
534         </plugin>
535     </plugins>
536 </build>
537
538 </project>
539
540
541 4. JUnit Test
542 1)src/test/java/com.example.demo.DemoApplicationTests.java > right-click > Run As > JUnit Test >
   Green bar
543
544 5. Spring Boot App 실행하기
545 1)demp project > right-click > Run As > Spring Boot App
546 2)http://localhost:8080/
547
548 Whitelabel Error Page
549 This application has no explicit mapping for /error, so you are seeing this as a fallback.
550
551 Thu Nov 07 15:47:32 KST 2019
552 There was an unexpected error (type=Not Found, status=404).
553 No message available
554
555
556 6. Controller 생성
557 1)src/main/java/com.example.demo > right-click > New > Class
558 2)Name : HelloController
559
560 package com.example.demo;
561

```

```
562 import org.springframework.web.bind.annotation.GetMapping;
563 import org.springframework.web.bind.annotation.RestController;
564
565 @RestController
566 public class HelloController {
567
568     @GetMapping("/")
569     public String hello() {
570         return "Hello, Spring Boot World";
571     }
572 }
573
574 7. Relaunch demo
575 -http://localhost:8080/
576
577 Hello, Spring Boot World
578
579 8. RestController 사용하기
580 1)HelloController.java 수정하기
581
582 @RestController
583 public class HelloController {
584
585     @GetMapping("/hello")
586     public String hello(String name) {
587         return "Hello! : " + name;
588     }
589 }
590
591 2)@RestController
592 -이 annotation을 붙이면 web application에서 request를 받아들이는 controller class임을 나타낸다.
593 -Spring 4부터 지원
594 -REST 방식의 응답을 처리하는 Controller를 구현할 수 있다.
595 -@Controller를 사용할 때 method의 return type이 문자열일 경우, 문자열에 해당하는 View를 만들어
596 야 하지만, Controller를 RestController를 사용할 경우에는 Return되는 문자열이 Browser에 그대로 출력
597 되기 때문에 별도로 View 화면을 만들 필요가 없다.
598
599 3)Relaunch demo
600 -http://localhost:8080/hello?name=한지민
601
602 Hello : 한지민
603
604 9. VO 사용하기
605 1)com.example.demo/DemoApplication.java 수정하기
606 @SpringBootApplication
607 @ComponentScan(basePackages = {"com.example"})
608 public class DemoApplication {
609     ...
610 }
611
```

```
612 2)com.example.vo package 생성
613 3)com.example.vo.UserVO Class 생성
614
615 package com.example.vo;
616
617 public class UserVO {
618     private String userid;
619     private String name;
620     private String gender;
621     private String city;
622
623     public UserVO() {}
624     public UserVO(String userid, String name, String gender, String city) {
625         this.userid = userid;
626         this.name = name;
627         this.gender = gender;
628         this.city = city;
629     }
630     public String getUserid() {
631         return userid;
632     }
633     public void setUserid(String userid) {
634         this.userid = userid;
635     }
636     public String getName() {
637         return name;
638     }
639     public void setName(String name) {
640         this.name = name;
641     }
642     public String getGender() {
643         return gender;
644     }
645     public void setGender(String gender) {
646         this.gender = gender;
647     }
648     public String getCity() {
649         return city;
650     }
651     public void setCity(String city) {
652         this.city = city;
653     }
654     @Override
655     public String toString() {
656         return "UserVO [userid=" + userid + ", name=" + name + ", gender=" + gender + ", city=" +
657             city + "]";
658     }
659 }
660
661 10. UserController 생성하기
662 1)com.example.controller package 생성
```

```

663 2)com.example.controller.UserController Class 생성
664
665     package com.example.controller;
666
667     import org.springframework.web.bind.annotation.GetMapping;
668     import org.springframework.web.bind.annotation.RestController;
669
670     import com.example.vo.UserVO;
671
672     @RestController
673     public class UserController {
674
675         @GetMapping("/getUser")
676         public UserVO getUser() {
677             UserVO user = new UserVO();
678             user.setUserid("jimin");
679             user.setName("한지민");
680             user.setGender("여");
681             user.setCity("서울");
682             return user;
683         }
684     }
685

```

```

686 3)Relaunch demo
687 -http://localhost:8080/getUser
688
689     {"userid":"jimin","name":"한지민","gender":"여","city":"서울"}
690
691

```

692 11. Spring DevTools 사용하기

- 693 1)위처럼 Controller에 새로운 Method가 추가되면 반드시 실행 중인 Application을 중지하고 Application을 재실행해야 한다.
- 694 2)그렇게 해야만 수정된 Controller가 반영되기 때문이다.
- 695 3)그렇게 반복적인 작업을 하지 않고, 즉 Controller가 수정할 때마다 매번 Application을 재실행하는 것이 번거로우면 Spring DevTools 기능을 이용하면 된다.
- 696 4)현재 사용중인 Project에 DevTools를 추가하려면 pom.xml에 추가 Dependency를 추가해야 한다.
- 697 5)pom.xml을 열어서 <dependency> 제일 마지막 Tag 밑에 Ctrl + Space 를 누른다.
- 698 6)Context Menu에서 [Edit Starters...]를 double-click한다.
- 699 7)[Pom file needs savings] 창에서 pom.xml 파일의 수정 내용을 저장할 것인지 물어보면 [Save Pom] button을 클릭한다.
- 700 8)[Edit Spring Boot Starters]창에서 Developer Tools > Spring Boot DevTools 체크한다.
- 701 9)그리고 [OK] 클릭한다.
- 702 10)그러면 pom.xml에 다음과 같은 Code가 추가된다.

```

703
704     <dependency>
705         <groupId>org.springframework.boot</groupId>
706         <artifactId>spring-boot-devtools</artifactId>
707         <scope>runtime</scope>
708     </dependency>
709

```

- 710 11)방금 추가된 DevTools 를 적용하기 위해 Application을 다시 실행한다.
- 711 12)Controller의 Code를 수정하면 자동으로 Restart가 일어난다.

712 13)Browser에서 Refresh를 누르면 수정된 Code가 반영된다.
713 14)즉, Java Code를 수정한 뒤 Application을 다시 수동으로 시작하지 않아도 된다.
714
715
716 12. Lombok Library 사용하기
717 1)보통 VO Class를 사용할 때 Table의 Column 이름과 같은 이름을 사용한다.
718 2)getter / setter method도 생성하고 toString() method도 생성한다.
719 3)하지만 code가 지저분해지고 모든 VO class와 JPA에서 사용할 Domain Class에 이런 Method를 반복적으로 작성하는 일은 사실 번거로운 일이다.
720 4)이런 문제를 간단하게 해결하기 위한 Library가 Lombok이다.
721 5)Lombok을 사용하면 Java File을 Compile할 때, 자동으로 생성자, getter / setter, toString() 같은 code들을 추가해준다.
722 6)현재 사용하고 있는 project에 Lombok Library를 추가해 보자.
723 7)위처럼 pom.xml의 <dependency> tag 제일 마지막에 Ctrl + Space 단축키를 누른다.
724 8)Context Menu에서 [Edit Starters...]를 double-click한다.
725 9)[Pom file needs savings] 창에서 pom.xml 파일의 수정 내용을 저장할 것인지 물어보면 [Save Pom] button을 클릭한다.
726 10)[Edit Spring Boot Starters]창에서 Developer Tools > Lombok 체크한다.
727 11)그리고 [OK] 클릭한다.
728 12)그러면 pom.xml에 다음과 같은 Code가 추가된다.
729
730 <dependency>
731 <groupId>org.projectlombok</groupId>
732 <artifactId>lombok</artifactId>
733 </dependency>
734
735 13)Lombok을 사용하려면 별도로 STS 설치 Folder에 Lombok Library를 추가해야 한다.
736 14)STS에 Lombok Library를 추가하기 위해 STS를 일단 종료한다.
737 15)Lombok Homepage(<https://projectlombok.org/>)를 방문한다.
738 16)download page로 이동하여 현재 최신 버전인 1.18.10을 Downloads 한다.
739 17)download 한 Folder로 이동하여 Cmd 창에서 아래의 명령을 수행한다.
740
741 java -jar lombok.jar
742
743 18)[Project Lombok v1.18.10 - Installer] 창에서, IDEs에 보면 현재 Eclipse와 STS가 설치된 folder가 자동 감지된다.
744 19)확인이 되었으면 [Install/Update] button click한다.
745 20)[Quit Installer] button click 한다.
746 21)Lombok이 설치되면 STS 설치 Folder(C:\Program Files\sts-4.4.0.RELEASE)에 lombok.jar가 있는 것을 확인할 수 있다.
747 22)다시 STS를 실행하여 UserVO.java로 들어간다.
748
749 package com.example.vo;
750
751 import lombok.Getter;
752 import lombok.Setter;
753 import lombok.ToString;
754
755 @Getter
756 @Setter
757 @ToString
758 public class UserVO {


```

759     private String userid;
760     private String name;
761     private String gender;
762     private String city;
763 }
764
765 23)수정된 UserVO.java 를 저장하고 왼쪽의 Package Explorer에서 UserVO.java의 하위를 클릭하면 Getter
/ Setter, toString() 이 자동으로 추가된 것을 확인할 수 있다.
766 24)다음은 Lombok에서 제공하는 Annotation이다.
767     -@Getter
768         -- Getter Method 생성
769     -@Setter
770         --Setter Method 생성
771     -@RequiredArgsConstructor
772         --모든 Member 변수를 초기화하는 생성자를 생성
773     -@ToString
774         --모든 Member 변수의 값을 문자열로 연결하여 리턴하는 toString() 메소드 생성
775     -@EqualsAndHashCode
776         --equals(), hashCode() Method 생성
777     -@Data
778         --@Getter, @Setter, @RequiredArgsConstructor, @ToString, @EqualsAndHashCode 모두 생성.
779
780
781
782 -----
783 Task5. SPRING INITIALIZER(Gradle)
784 1. Visit http://start.spring.io/
785 2. 설정
786     1)Gradle Project
787     2)Java
788     3)2.2.1
789     4)Group : com.example
790     5)Artifact : demoweb
791     6)Name : demoweb
792     7)Description : Demo project for Spring Boot
793     8)Package Name : com.example.demoweb
794     9)Packaging : Jar
795     10)Java Version : 8
796     11)dependencies : Developer Tools > SpringBoot DevTools, Web > Web
797
798     12)Click [Generate]
799     13)Downloads [demoweb.zip] : 57.8KB
800     14)Unpack to Spring workspace.
801
802
803 3. Project Import
804     1)In STS, Package Explorer > right-click > Import > Gradle > Existing Gradle Project > Next > Next
805     2)Click [Browse...] > demoweb Folder > Select Folder > Next > Finish
806     3)build.gradle
807
808     plugins {
809         id 'org.springframework.boot' version '2.2.1.RELEASE'

```

```
810     id 'io.spring.dependency-management' version '1.0.8.RELEASE'
811     id 'java'
812 }
813
814 group = 'com.example'
815 version = '0.0.1-SNAPSHOT'
816 sourceCompatibility = '1.8'
817
818 configurations {
819     developmentOnly
820     runtimeClasspath {
821         extendsFrom developmentOnly
822     }
823 }
824
825 repositories {
826     mavenCentral()
827 }
828
829 dependencies {
830     implementation 'org.springframework.boot:spring-boot-starter-web'
831     developmentOnly 'org.springframework.boot:spring-boot-devtools'
832     testImplementation('org.springframework.boot:spring-boot-starter-test') {
833         exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
834     }
835 }
836
837 test {
838     useJUnitPlatform()
839 }
840
841
842 4)src/test/java/com.example.demoweb.DemowebApplicationTests.java > right-click > Run As > JUnit
Test > Green bar
843 5)demoweb Project > right-click > Run As > Spring Boot App
844 6)http://localhost:8080/
845
846 Whitelabel Error Page
847 This application has no explicit mapping for /error, so you are seeing this as a fallback.
848
849 Thu Nov 07 16:06:13 KST 2019
850 There was an unexpected error (type=Not Found, status=404).
851 No message available
852
853
854 4. Controller 생성
855 1)src/main/java/com.example.demoweb > right-click > New > Class
856 2)Name : HomeController
857
858 package com.example.demo;
859
860 import org.springframework.web.bind.annotation.GetMapping;
```

```
861 import org.springframework.web.bind.annotation.RestController;
862
863 @RestController
864 public class HomeController {
865
866     @GetMapping("/")
867     public String home() {
868         return "Hello, Spring Boot World";
869     }
870 }
871
872 3)Relaunch demo
873 4)http://localhost:8080/
874
875 Hello, Spring Boot World
876
877 -----
878 Task6. 사용자 정의 Starter 만들기
879 1. Maven Project 생성
880 1)Project Explorer > right-click > New > Maven Project
881 2)Next
882 3)org.apache.maven.archetypes, maven-archetype-quickstart, 1.1 > Next
883 4)Group Id : com.example
884     -Artifact Id : mybootstarter
885     -Version : 0.0.1-SNAPSHOT
886     -Package : com.example.mybootstarter
887     -Finish
888
889 2. Project Facets 수정하기
890 1)mybootstarter Project > right-click > Properties > Project Facets
891 2)Java 1.8 > Runtimes Tab > Apache Tomcat v9.0, jdk1.8.0_221 check
892 3)Apply and Close click
893
894 3. Mvnrepository에서 'spring boot'로 검색
895 1)Spring Boot Autoconfigure > 2.2.0.RELEASE
896 2)아래 코드 복사 후 pom.xml 에 붙여넣기
897
898
899 <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-autoconfigure -->
900 <dependency>
901     <groupId>org.springframework.boot</groupId>
902     <artifactId>spring-boot-autoconfigure</artifactId>
903     <version>2.2.0.RELEASE</version>
904 </dependency>
905
906 3)pom.xml > right-click > Run As > Maven install
907 [INFO] BUILD SUCCESS
908
909 4. dependencyManagement 추가
910 1)앞으로 추가되는 Library들의 Version을 일괄적으로 관리하기 위해 pom.xml에 Code 추가
911 2)Mvnrepository에서 'spring boot dependencies'로 검색
912 3)Spring Boot Dependencies에서 2.2.0.RELEASE
```

```
913 4)아래의 Code를 pom.xml의 제일 아래에 추가
914
915 <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-dependencies -->
916 <dependency>
917     <groupId>org.springframework.boot</groupId>
918     <artifactId>spring-boot-dependencies</artifactId>
919     <version>2.2.0.RELEASE</version>
920     <type>pom</type>
921     <scope>provided</scope>
922 </dependency>
923
924 5)pom.xml
925
926 <dependencyManagement>
927     <dependencies>
928         <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-dependencies -->
929         <dependency>
930             <groupId>org.springframework.boot</groupId>
931             <artifactId>spring-boot-dependencies</artifactId>
932             <version>2.2.0.RELEASE</version>
933             <type>pom</type>
934             <scope>provided</scope>
935         </dependency>
936     </dependencies>
937 </dependencyManagement>
938
939 6)pom.xml > right-click > Run As > Maven install
940 [INFO] BUILD SUCCESS
941
942
943 5. 자동설정 구현하기
944 1)이번 실습은 JDBC로 직접 Database 연동을 처리하는 Application을 위한 자동 설정이 목표이다.
945 2)com.example.util package 생성
946 3)com.example.util.JdbcConnectionManager.java 구현
947
948 package com.example.util;
949
950 import java.sql.Connection;
951 import java.sql.DriverManager;
952
953 public class JdbcConnectionManager {
954     private String driverClass;
955     private String url;
956     private String username;
957     private String password;
958
959     public void setDriverClass(String driverClass) {
960         this.driverClass = driverClass;
961     }
962     public void setUrl(String url) {
963         this.url = url;
964     }
965 }
```

```

965     public void setUsername(String username) {
966         this.username = username;
967     }
968     public void setPassword(String password) {
969         this.password = password;
970     }
971
972     public Connection getConnection() {
973         try {
974             Class.forName(this.driverClass);
975             return DriverManager.getConnection(this.url, this.username, this.password);
976         } catch (Exception e) {
977             e.printStackTrace();
978         }
979         return null;
980     }
981     @Override
982     public String toString() {
983         return "JdbcConnectionManager [driverClass=" + driverClass + ", url=" + url + ", username=" +
984             + ", password=" + password + "];"
985     }
986
987 }

```

4)JdbcConnectionManager를 bean으로 등록하는 환경 설정 Class를 작성한다.

5)com.example.config package 생성

6)com.example.config.UserAutoConfiguration.java 생성

```

992
993     package com.example.config;
994
995     import org.springframework.context.annotation.Bean;
996     import org.springframework.context.annotation.Configuration;
997
998     import com.example.util.JdbcConnectionManager;
999
1000     @Configuration
1001     public class UserAutoConfiguration {
1002
1003         @Bean
1004         public JdbcConnectionManager getJdbcConnectionManager() {
1005             JdbcConnectionManager manager = new JdbcConnectionManager();
1006             manager.setDriverClass("oracle.jdbc.driver.OracleDriver");
1007             manager.setUrl("jdbc:oracle:thin:@localhost:1521:XE");
1008             manager.setUsername("hr");
1009             manager.setPassword("hr");
1010             return manager;
1011         }
1012     }
1013

```

1014 6. src/main/resources folder 추가

1015 1)mybootstarter project > right-click > New > Source Folder

1016 2)Folder name : src/main/resources
1017 3)Finish
1018
1019 7. resources/META-INF folder 생성
1020 1)src/main/resources > right-click > New > Folder
1021 2)Folder name : META-INF
1022 3)Finish
1023
1024 8. src/main/resources/META-INF/spring.factories file 생성
1025
1026 org.springframework.boot.autoconfigure.EnableAutoConfiguration=#com.example.config.UserAutoConf
1027 uration
1028 9. Build
1029 1)mybootstarter project > right-click > Run As > Maven install
1030 [INFO] BUILD SUCCESS
1031 2)성공하면 C:\Users\bluee\m2\repository\com\example\mybootstarter\0.0.1-SNAPSHOT에
1032 packaging 한 jar 파일이 등록되어 있을 것이다.
1033
1034 10. Starter와 자동 설정 사용하기
1035 1)사용자 정의 Starter가 Maven Repository에 등록됐으면 이제 이 Starter를 이용하여 Application을 만들
1036 수 있다.
1037 2)위에서 생성한 Project의 pom.xml에서 아래의 Code를 복사한다.
1038 <groupId>com.example</groupId>
1039 <artifactId>mybootstarter</artifactId>
1040 <version>0.0.1-SNAPSHOT</version>
1041
1042 3)위의 Task4 에서 생성한 demo Project의 pom.xml의 <dependency>에 붙여넣는다.
1043
1044 <dependency>
1045 <groupId>org.projectlombok</groupId>
1046 <artifactId>lombok</artifactId>
1047 </dependency>
1048
1049 <!-- 아래 코드 추가 -->
1050 <dependency>
1051 <groupId>com.example</groupId>
1052 <artifactId>mybootstarter</artifactId>
1053 <version>0.0.1-SNAPSHOT</version>
1054 </dependency>
1055
1056 4)pom.xml > right-click > Run As > Maven install
1057 [INFO] BUILD SUCCESS
1058
1059 5)위와 같이 BUILD SUCCESS가 나오면, demo Project의 Maven Dependencies의 목록에 보면
1060 mybootstarter가 추가된 것을 확인할 수 있다.
1061 6)이제 demo Project에 추가된 mybootstarter 를 사용하는 프로그래밍을 작성한다.
1062 7)demo Project에 com.example.service package를 생성한다.
1063 8)com.example.service.JdbcConnectionManagerRunner Class를 생성한다.

```

1064 package com.example.service;
1065
1066 import org.springframework.beans.factory.annotation.Autowired;
1067 import org.springframework.boot.ApplicationArguments;
1068 import org.springframework.boot.ApplicationRunner;
1069 import org.springframework.stereotype.Service;
1070
1071 import com.example.util.JdbcConnectionManager;
1072
1073 @Service
1074 public class JdbcConnectionManagerRunner implements ApplicationRunner {
1075     @Autowired
1076     private JdbcConnectionManager connectionManager;
1077
1078     @Override
1079     public void run(ApplicationArguments args) throws Exception {
1080         System.out.println("Connection Manager : " + this.connectionManager.toString());
1081     }
1082 }
1083
1084 9)위에서 생선한 JdbcConnectionManagerRunner는 Container가 Component Scan하도록 @Service를 추
가했다.
1085 10)ApplicationRunner Interface를 구현했기 때문에 JdbcConnectionManagerRunner 객체가 생성되자마자
Container에 의해서 run() 가 자동으로 실행된다.
1086 11)demo Project > right-click > Run As > Spring Boot App
1087 12)Console에 다음과 같은 출력이 나오면 자동설정이 정상적으로 동작했다는 의미이다.
1088
1089 Connection Manager : JdbcConnectionManager [driverClass=oracle.jdbc.driver.OracleDriver,
url=jdbc:oracle:thin:@localhost:1521:XE, username=hr, password=hr]
1090
1091 13)만일 자동설정이 동작하지 않았다면 의존성 주입에서 Error가 발생했을 것이다.
1092
1093
1094 11. 자동설정 재정의하기
1095 1)Bean 재정의하기
1096 -현재 mybootstarter는 Oracle과 Connection을 할 수 있다.
1097 -이것을 MariaDB로 변경하려고 한다.
1098 -demo Project에 com.example.config package 생성한다.
1099 -com.example.config.UserConfiguration Class를 생성한다.
1100
1101 package com.example.config;
1102
1103 import org.springframework.context.annotation.Bean;
1104 import org.springframework.context.annotation.Configuration;
1105
1106 import com.example.util.JdbcConnectionManager;
1107
1108 @Configuration
1109 public class UserConfiguration {
1110     @Bean
1111     public JdbcConnectionManager getJdbcConnectionManager() {
1112         JdbcConnectionManager manager = new JdbcConnectionManager();

```

```

1113         manager.setDriverClass("org.mariadb.jdbc.Driver");
1114         manager.setUrl("jdbc:mariadb://localhost:3306/test");
1115         manager.setUsername("root");
1116         manager.setPassword("javamariadb");
1117         return manager;
1118     }
1119 }

```

-이렇게 하면 자동 설정으로 등록한 bean을 새로 등록한 bean이 덮어쓰면서 Oracle에서 MariaDB의 JdbcConnectionManager를 사용할 수 있다.

-그런데, Application을 실행해 보면 Console에는 Error가 발생한다.

```

1124 The bean 'getJdbcConnectionManager', defined in class path resource
[com/example/config/UserAutoConfiguration.class], could not be registered. A bean with that
name has already been defined in class path resource
[com/example/config/UserConfiguration.class] and overriding is disabled.

```

-즉, Memory에 같은 Type의 bean이 두 개가 등록되어 충돌이 발행했다는 메시지이다.

-이 문제를 해결하기 위해서는 새로 생성된 bean이 기존에 등록된 bean을 덮어쓸 수 있도록 해야 한다.

-demo Project의 src/main/resources/application.properties 파일에 다음의 설정을 추가한다.

```

1130 ## Bean Overriding 설정
1131 spring.main.allow-bean-definition-overriding=true

```

※Properties Editor Plugin 설치하기

1. application.properties 파일에 작성한 한글이 정상적으로 보이지 않으면 [Properties Editor] plugin을 설치하면 된다.
2. Help > Install New Software... > Add...
3. Name : Properties Editor
4. Location : <http://propedit.sourceforge.jp/eclipse/updates>
5. Add
6. 이렇게 설치하는 이유는 현재 Eclipse Marketplace에서 'Properties Editor'로 검색되지 않기 때문이다.
7. 목록에서 [PropertiesEditor]만 Check하고 Next
8. 다른 Plugin 설치와 마찬가지로 계속 설치를 진행한다.
9. 설치과정이 마치면 STS를 재 시작하고 application.properties file을 선택하고 Mouse right-click > Open With > PropertiesEditor
10. 한글이 깨지지 않고 정상적으로 보이는 것을 볼 수 있다.

-demo Project를 다시 실행하면 Error는 나오지 않는데, 아직도 Oracle의 설정 정보가 나오는 것을 볼 수 있다.

-그 이유는 demo Project의 bean으로 등록한 JdbcConnectionManager가 사용된 것이 아니라 mybootstarter Project에서 등록한 JdbcConnectionManager를 사용했기 때문이다.

-이 문제를 해결하기 위한 Annotation이 바로 @Conditional이다.

-[@Conditional](#) Annotation은 조건에 따라 새로운 객체를 생성할지 안할지를 결정할 수 있다.

2)@Conditional Annotation 사용하기

-[@SpringBootApplication](#)은 @EnableAutoConfiguration과 @ComponentScan을 포함하고 있다.

-Spring Boot는 @ComponentScan을 먼저 처리하여 사용자가 등록한 Bean을 먼저 Memory에 올린다.

-그리고 나중에 @EnableAutoConfiguration을 실행하여 자동 설정에 위한 Bean 등록을 처리한다.

-따라서 위에서 새로 생성한 Bean(MariaDB용 Connector)을 자동 설정한 Bean(Oracle Connector)이 덮어버린 것이다.

-mybootstarter Project의 com.example.config.UserAutoConfiguration Class에

@ConditionalOnMissingBean Annotation을 적용한다.

```

1156
1157 package com.example.config;
1158
1159 import org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
1160 import org.springframework.context.annotation.Bean;
1161 import org.springframework.context.annotation.Configuration;
1162
1163 import com.example.util.JdbcConnectionManager;
1164
1165 @Configuration
1166 public class UserAutoConfiguration {
1167
1168     @Bean
1169     @ConditionalOnMissingBean
1170     public JdbcConnectionManager getJdbcConnectionManager() {
1171         JdbcConnectionManager manager = new JdbcConnectionManager();
1172         manager.setDriverClass("oracle.jdbc.driver.OracleDriver");
1173         manager.setUrl("jdbc:oracle:thin:@localhost:1521:XE");
1174         manager.setUsername("hr");
1175         manager.setPassword("hr");
1176         return manager;
1177     }
1178 }
1179
1180

```

-[@ConditionalOnMissingBean](#)은 등록하려는 Bean이 Memory에 없는 경우에만 현재의 Bean 등록을 처리하도록 한다.

-따라서 사용자가 정의한 JdbcConnectionManager Bean이 @ComponentBean 설정에 의해 먼저 등록된다면 자동설정인 @EnableAutoConfiguration이 동작하는 시점에는 이미 등록된 Bean을 사용하고 새롭게 Bean을 생성하지 않는다.

-이제 모두 저장하고 mybootstarter Project를 다시 install한다.

--mybootstarter Project > right-click > Run As > Maven install

[INFO] BUILD SUCCESS

-그리고 demo Project를 Refresh하고 다시 Project를 실행하면 다음과 같이 Oracle에서 MariaDB로 변경된 것을 알 수 있다.

```

Connection Manager : JdbcConnectionManager [driverClass=org.mariadb.jdbc.Driver,
url=jdbc:mariadb://localhost:3306/test, username=root, password=javamariadb]

```

3)Property File 이용하기

-Spring Container가 생성한 Bean의 Member Variable의 값이 자주 변경된다면 변경될 때마다 Java Source를 수정하기 보다 변경되는 정보만 Property로 등록하고 이 Property 정보를 이용해서 Bean을 생성하면 편리하다.

-Lab을 위해서 demo Project의 com.example.config.UserConfiguration.java의 @Configuration과 @Bean을 주석처리한다.

```

1194
1195 //@Configuration
1196 public class UserConfiguration {
1197     //@Bean
1198     public JdbcConnectionManager getJdbcConnectionManager() {

```

```
1199         JdbcConnectionManager manager = new JdbcConnectionManager();
1200         manager.setDriverClass("org.mariadb.jdbc.Driver");
1201         manager.setUrl("jdbc:mariadb://localhost:3306/test");
1202         manager.setUsername("root");
1203         manager.setPassword("javamariadb");
1204         return manager;
1205     }
1206 }
```

1207
1208 -demo Project의 application.properties 파일에 다음 코드를 추가한다.

```
1209  
1210     ## Bean Overriding 설정
1211     spring.main.allow-bean-definition-overriding=true
1212  
1213     ## 데이터 소스 : Oracle
1214     user.jdbc.driverClass=oracle.jdbc.driver.OracleDriver
1215     user.jdbc.url=jdbc:oracle:thin:@localhost:1521:XE
1216     user.jdbc.username=hr
1217     user.jdbc.password=hr
1218
```

1219 -mybootstarter Project의 com.example.util.JdbcConnectionManagerProperties Class 추가로 생성한다.
1220 -이미 만들어 놓은 com.example.util.JdbcConnectionManager.java를 복사, 붙여넣기, 이름변경을
JdbcConnectionManagerProeprties로 한다.

```
1221  
1222     package com.example.util;
1223  
1224     import java.sql.Connection;
1225     import java.sql.DriverManager;
1226  
1227     import org.springframework.boot.context.properties.ConfigurationProperties;
1228  
1229     @ConfigurationProperties(prefix="user.jdbc")
1230     public class JdbcConnectionManagerProperties {
1231         private String driverClass;
1232         private String url;
1233         private String username;
1234         private String password;
1235  
1236         public String getDriverClass() {
1237             return driverClass;
1238         }
1239         public void setDriverClass(String driverClass) {
1240             this.driverClass = driverClass;
1241         }
1242         public String getUrl() {
1243             return url;
1244         }
1245         public void setUrl(String url) {
1246             this.url = url;
1247         }
1248         public String getUsername() {
1249             return username;
```

```
1250     }
1251     public void setUsername(String username) {
1252         this.username = username;
1253     }
1254     public String getPassword() {
1255         return password;
1256     }
1257     public void setPassword(String password) {
1258         this.password = password;
1259     }
1260 }
```

1262 -위와 같이 작성하고 저장하면 노란색 경로라인이 발생한다.

1263 -노란색 경고 메시지에 마우스를 올려놓으면 Add spring-boot-configuration-processor to pom.xml link를 click한다.

1264 -이렇게 하면 자동으로 pom.xml에 다음과 같은 dependency가 추가된다.

```
1266     <dependency>
1267         <groupId>org.springframework.boot</groupId>
1268         <artifactId>spring-boot-configuration-processor</artifactId>
1269         <optional>true</optional>
1270     </dependency>
```

1272 -Error를 방지하기 위해 <version>2.2.0.RELEASE</version>을 추가한다.

1273 -mybootstarter Project > right-click > Maven > Update Project > OK

1274 -pom.xml > right-click > Run As > Maven install

1275 [INFO] BUILD SUCCESS

1277 -이제 mybootstarter Project의 com.example.config.UserAutoConfiguration Class를 수정한다.

```
1279 package com.example.config;
```

```
1281 import org.springframework.beans.factory.annotation.Autowired;
1282 import org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
1283 import org.springframework.boot.context.properties.EnableConfigurationProperties;
1284 import org.springframework.context.annotation.Bean;
1285 import org.springframework.context.annotation.Configuration;
```

```
1287 import com.example.util.JdbcConnectionManager;
1288 import com.example.util.JdbcConnectionManagerProperties;
```

```
1290 @Configuration
```

```
1291 @EnableConfigurationProperties(JdbcConnectionManagerProperties.class)
```

```
1292 public class UserAutoConfiguration {
```

```
1294     @Autowired
```

```
1295     private JdbcConnectionManagerProperties properties;
```

```
1297     @Bean
```

```
1298     @ConditionalOnMissingBean
```

```
1299     public JdbcConnectionManager getJdbcConnectionManager() {
```

```
1300         JdbcConnectionManager manager = new JdbcConnectionManager();
```

```

1301         manager.setDriverClass(this.properties.getDriverClass());
1302         manager.setUrl(this.properties.getUrl());
1303         manager.setUsername(this.properties.getUsername());
1304         manager.setPassword(this.properties.getPassword());
1305         return manager;
1306     }
1307 }
1308
1309 -이제 mybootstarter Project를 다시 Install 한다.
1310 --mybootstarter Project > right-click > Run As > Maven install
1311 [INFO] BUILD SUCCESS
1312 -demo Project를 Refresh하고 다시 실행한다.
1313 --다시 Oracle 설정 정보가 나오는 것을 알 수 있다.
1314
1315 Connection Manager : JdbcConnectionManager [driverClass=oracle.jdbc.driver.OracleDriver,
1316 url=jdbc:oracle:thin:@localhost:1521:XE, username=hr, password=hr]
1317
1318 -만일 Database가 다시 MaraiDB로 변경된다면 demo Project의 application.properties를 다음과 같이
1319 변경하면 된다.
1320
1321 ## Bean Overriding 설정
1322 spring.main.allow-bean-definition-overriding=true
1323
1324 ## 데이터 소스 : Oracle
1325 #user.jdbc.driverClass=oracle.jdbc.driver.OracleDriver
1326 #user.jdbc.url=jdbc:oracle:thin:@localhost:1521:XE
1327 #user.jdbc.username=hr
1328 #user.jdbc.password=hr
1329
1330 ## 데이터 소스 : MariaDB
1331 user.jdbc.driverClass=org.mariadb.jdbc.Driver
1332 user.jdbc.url=jdbc:mariadb://localhost:3306/test
1333 user.jdbc.username=root
1334 user.jdbc.password=javamariadb
1335
1336 -저장하면 바로 MaraiDB로 설정정보가 변경된 것을 알 수 있다.
1337
1338 Connection Manager : JdbcConnectionManager [driverClass=org.mariadb.jdbc.Driver,
1339 url=jdbc:mariadb://localhost:3306/test, username=root, password=javamariadb]
1340 -----
1341 Task7. 간단한 JPA Project
1342 1. H2 Database 설치하기
1343 -여러 RDBMS가 있지만 H2를 사용하려는 이유는 Spring Boot가 기본적으로 H2를 지원하고 있기 때문이다.
1344 -H2는 Java로 만들어졌으며, 용량이 작고 실행 속도가 빠른 Open Source Database이다.
1345 -H2 Homepage(http://www.h2database.com/html/main.html)를 방문한다.
1346 -Download의 All Platforms Link를 Click하여 압축파일을 Download한다.
1347 -압축을 풀고 h2 Folder를 C:/Program Files로 이동한다.
1348 -h2/bin의 h2w.bat를 실행하면 Browser기반의 관리 Console이 열린다.

```

1349 -Tray에 있는 h2 Console을 실행하여 다음의 각 항목에 값을 입력하고 [Create] button을 click한다.
1350 --Database path : ./test
1351 --Username : sa
1352 --Password : javah2
1353 --Password confirmation : javah2
1354 --Create button click
1355
1356 -----
1357 Database was created successfully.
1358
1359 JDBC URL for H2 Console:
1360 jdbc:h2:./test
1361
1362 -각 항목의 정보를 입력하고 [Test Connection] 클릭해본다.
1363 --Driver Class : org.h2.Driver
1364 --JDBC URL : jdbc:h2:tcp://localhost/~./test
1365 --User Name : sa
1366 --Password : javah2
1367
1368 -연결이 성공하면 Web Console이 열린다.
1369
1370
1371 2. JPA Project Installation
1372 1)Help > Install New Software
1373 2)Work with : <https://download.eclipse.org/releases/2019-09>
1374 3)Filter : jpa
1375 4)결과에서
1376 Web, XML, Java EE and OSGi Enterprise Development하위의
1377 -Dali Java Persistence Tools - EclipseLink JPA Support
1378 -Dali Java Persistence Tools - JPA Diagram Editor
1379 -Dali Java Persistence Tools - JPA Support
1380 -m2e-wtp - JPA configurator for WTP (Optional)
1381 5)설치 후 STS Restart
1382
1383
1384 3. JPA Project 생성
1385 1)Package Explorer > right-click > Maven Project
1386 -Next
1387 -org.apache.maven.archetypes, maven-archetype-quickstart, 1.1
1388 -Next
1389 2)각 항목 선택 후 Finish
1390 -Group Id : com.example
1391 -artifact Id : jpademo
1392 -Version : 0.0.1-SNAPSHOT
1393 -Package : com.example.jpademo
1394
1395 3)Project Facets 변환
1396 -jpademo Project > right-click > Properties > Project Facets
1397 -Java 1.8 > Runtimes > Apache Tomcat v9.0, jdk1.8.0_221 Check
1398 -Check JPA
1399 -만일 설정 화면 하단의 [Further configuration required...] Link에 Error message가 뜨는 경우
1400 --Link click

1401 --[JPA Facet] 창에서
1402 ---Platform : Generic 2.1
1403 ---JPA implementation
1404 Type : Disable Library Configuration
1405 --OK
1406 -Apply and Close
1407
1408 4)Maven Project를 JPA Project로 변경하면 src/main/java하위에 META-INF/persistence.xml JPA 환경설정
파일이 생긴다.
1409 5)jpademo Project의 Perspective를 JPA Perspective로 변경하려면
1410 -Window > Perspective > Open Perspective > Other
1411 -JPA 선택 > Open
1412
1413
1414 4. 의존성 추가
1415 1)pom.xml을 수정
1416 -Mvnrepository에서 'hibernate'로 검색하여 'Hibernate ORM Hibernate EntityManager'로 들어간다.
1417 -5.4.8.Final Click
1418
1419 <!-- <https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager> -->
1420 <dependency>
1421 <groupId>org.hibernate</groupId>
1422 <artifactId>hibernate-entitymanager</artifactId>
1423 <version>5.4.8.Final</version>
1424 </dependency>
1425
1426 -'h2'로 검색하여 'H2 Database Engine'으로 들어가서 1.4.200 선택
1427
1428 <!-- <https://mvnrepository.com/artifact/com.h2database/h2> -->
1429 <dependency>
1430 <groupId>com.h2database</groupId>
1431 <artifactId>h2</artifactId>
1432 <version>1.4.200</version>
1433 <!--<scope>test</scope> --> 주의할 것, 이 scope tag는 반드시 삭제할 것
1434 </dependency>
1435
1436 -pom.xml > right-click > Maven install
1437 [INFO] BUILD SUCCESS
1438
1439
1440 5. Entity Class 작성 및 Table Mapping
1441 1)Table을 준비한다.
1442 2)JPA는 Table이 없으면 Java Class를 기준으로 Mapping할 Table을 자동으로 생성한다.
1443 3)Table과 Mapping되는 Java Class를 Entity라고 한다.
1444 4)JPA를 사용하는 데 있어서 가장 먼저 해야 할 일은 Entity를 생성하는 것이다.
1445 5)Value Object Class처럼 Table과 동일한 이름을 사용하고 Column과 Mapping 될 Member Variable을 선
연하면 된다.
1446 6)다만, Eclipse의 JPA Perspective가 제공하는 Entity 생성 기능을 사용하면 Entity를 생성함과 동시에 영속
성 설정 파일(persistence.xml)에 자동으로 Entity가 등록된다.
1447 7)src/main/java Folder에 ocm.example.jpademo package > right-click > New > JPA Entity
1448 8)다음 항목의 값을 입력 후 Finish 클릭
1449 -Java package : com.example.domain

```
1450     -Class name : User
1451
1452 9)META-INF/persistence.xml 파일이 자동으로 수정되었다.
1453
1454     <?xml version="1.0" encoding="UTF-8"?>
1455     <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">
1456         <persistence-unit name="jpademo">
1457             <class>com.example.domain.User</class>
1458         </persistence-unit>
1459     </persistence>
1460
1461 10)이제 방금 생성한 User Class를 수정한다.
1462
1463     package com.example.domain;
1464
1465     import javax.persistence.Entity;
1466     import javax.persistence.Id;
1467     import javax.persistence.Table;
1468
1469     @Entity
1470     @Table
1471     public class User{
1472         @Id
1473         private String userid;
1474         private String username;
1475         private String gender;
1476         private int age;
1477         private String city;
1478         public String getUserid() {
1479             return userid;
1480         }
1481         public void setUserid(String userid) {
1482             this.userid = userid;
1483         }
1484         public String getUsername() {
1485             return username;
1486         }
1487         public void setUsername(String username) {
1488             this.username = username;
1489         }
1490         public String getGender() {
1491             return gender;
1492         }
1493         public void setGender(String gender) {
1494             this.gender = gender;
1495         }
1496         public int getAge() {
1497             return age;
1498         }
1499     }
```

```

1499     public void setAge(int age) {
1500         this.age = age;
1501     }
1502     public String getCity() {
1503         return city;
1504     }
1505     public void setCity(String city) {
1506         this.city = city;
1507     }
1508     @Override
1509     public String toString() {
1510         return "User [userid=" + userid + ", username=" + username + ", gender=" + gender + ",
1511             age=" + age + ", city="
1512             + city + "]\n";
1513     }
1514 }
1515

```

11)다음은 JPA 사용하는 주요 Annotation을 설명한 것이다.

-@Entity

--Entity Class 임을 설명

--기본적으로 Class의 이름과 동일한 Table과 Mapping된다.

-@Table

--Entity의 이름과 Table의 이름이 다를 경우, name 속성을 이용하여 Mapping 한다.

--이름이 동일하면 생략 가능

-@Id

--Table의 primary key와 Mapping한다.

--Entity의 필수 Annotation으로서 @Id가 없으면 Entity는 사용 불가

-@GeneratedValue

--@Id가 선언된 Field에 기본 키 값을 자동으로 할당

6. JPA main 설정 파일 작성

1)META-INF/persistence(JPA의 main 환경설정 파일) 수정

```

1533 <?xml version="1.0" encoding="UTF-8"?>
1534 <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">
1535     <persistence-unit name="jpademo">
1536         <class>com.example.domain.User</class>
1537
1538         <properties>
1539             <!-- 필수 속성 -->
1540             <property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>
1541             <property name="javax.persistence.jdbc.user" value="sa"/>
1542             <property name="javax.persistence.jdbc.password" value="javah2"/>
1543             <property name="javax.persistence.jdbc.url" value="jdbc:h2:tcp://localhost/~:/test"/>
1544             <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
1545
1546             <!-- Option 속성 -->

```



```

1547     <property name="hibernate.show_sql" value="true"/>
1548     <property name="hibernate.format_sql" value="true"/>
1549     <property name="hibernate.use_sql_comments" value="false"/>
1550     <property name="hibernate.id.new_generator_mappings" value="true"/>
1551     <property name="hibernate.hbm2ddl.auto" value="create"/>
1552 </properties>
1553 </persistence-unit>
1554 </persistence>
1555

```

2)여기서 중요한 속성은 hibernate.dialect 이다.

3)이 속성은 JPA 구현체가 사용할 Dialect Class를 지정할 때 사용한다.

4)이 속성을 H2Dialect Class로 설정하면 H2용 SQL이 생성되고, OracleDialect로 변경하면 Oracle용 SQL이 생성된다.

1559
1560 7. JPA로 Data 처리하기

1561 1)User 등록

1562 -src/main/java Folder에 JPAClient Class를 생성한다.

```

1563
1564     import javax.persistence.EntityManager;
1565     import javax.persistence.EntityManagerFactory;
1566     import javax.persistence.Persistence;
1567
1568     import com.example.domain.User;
1569
1570     public class JPAClient {
1571     public static void main(String[] args) {
1572         //EntityManager 생성
1573         EntityManagerFactory emf =
1574             Persistence.createEntityManagerFactory("jpademo");
1575         EntityManager em = emf.createEntityManager();
1576         try {
1577             User user = new User();
1578             user.setUserid("jimin");
1579             user.setUsername("한지민");
1580             user.setGender("여");
1581             user.setAge(24);
1582             user.setCity("서울");
1583
1584             //등록
1585             em.persist(user);
1586         }catch(Exception e) {
1587             e.printStackTrace();
1588         }finally {
1589             em.close();
1590             emf.close();
1591         }
1592     }
1593 }
1594

```

1595 -JPA는 META-INF/persistence.xml을 먼저 loading한다.

1596 -그리고 persistence.xml의 unit name으로 설정한 영속성 unit 정보를 이용하여 EntityManagerFactory 객체를 생성한다.

```
1597     -JPA를 이용하여 CRUD를 하려면 EntityManager 객체를 사용해야 한다.
1598     -이것은 EntityManagerFactory를 통해 생성되며, EntityManager를 얻었으면 persist() 를 통해 User
1599     Table에 저장한다.
1600 2)실행하기
1601     -JPAClient > right-click > Run As > Java Application
1602
1603 3)실행 결과
1604
1605     Hibernate:
1606
1607     drop table User if exists
1608
1609     Hibernate:
1610
1611     drop sequence if exists hibernate_sequence
1612     Hibernate: create sequence hibernate_sequence start with 1 increment by 1
1613
1614     Hibernate:
1615
1616     create table User (
1617         userid varchar(255) not null,
1618         age integer not null,
1619         city varchar(255),
1620         gender varchar(255),
1621         username varchar(255),
1622         primary key (userid)
1623     )
1624
1625 4)하지만 실제 Database에는 Data가 Insert되지 않았다.
1626
1627
1628 8. Transaction 관리
1629 1)JPA가 실제 Table에 등록/수정/삭제 작업을 처리하기 위해서는 해당 작업이 반드시 Transaction안에서
1630 수행되어야 한다.
1631 2)만약 Transaction을 시작하지 않았거나 등록/수정/삭제 작업 이후에 Transaction을 종료하지 않으면 요
1632 청한 작업이 실제 Database에 반영되지 않는다.
1633 3)JPAClient.java를 수정한다.
1634
1635     import javax.persistence.EntityManager;
1636     import javax.persistence.EntityManagerFactory;
1637     import javax.persistence.EntityTransaction;
1638     import javax.persistence.Persistence;
1639
1640     import com.example.domain.User;
1641
1642     public class JPAClient {
1643         public static void main(String[] args) {
1644             //EntityManager 생성
1645             EntityManagerFactory emf =
1646                 Persistence.createEntityManagerFactory("jpademo");
1647             EntityManager em = emf.createEntityManager();
```

```

1646
1647     //Transaction 생성
1648     EntityTransaction tx = em.getTransaction();
1649     try {
1650         //Transaction 시작
1651         tx.begin();
1652
1653         User user = new User();
1654         user.setUserid("jimin");
1655         user.setUsername("한지민");
1656         user.setGender("여");
1657         user.setAge(24);
1658         user.setCity("서울");
1659
1660         //등록
1661         em.persist(user);
1662
1663         // Transaction commit
1664         tx.commit();
1665     }catch(Exception e) {
1666         e.printStackTrace();
1667         // Transaction rollback
1668         tx.rollback();
1669     }finally {
1670         em.close();
1671         emf.close();
1672     }
1673 }
1674 }

```

4)실행하기

-JPAClient > right-click > Run As > Java Application

Hibernate:

```

insert
into
    User
    (age, city, gender, username, userid)
values
    (?, ?, ?, ?, ?)

```

5)H2 Database Console에서 Run을 수행하면 방금 입력한 데이터가 삽입된 것을 볼 수 있다.

9. 데이터 누적하기

1)현재 작성한 JPA 프로그램은 아무리 많이 실행해도 한 건의 Data만 등록된다.

2)즉 매번 Table이 새롭게 생성되기 때문이다.

3)따라서 JPA Client를 실행할 때마다 Data를 누적하기 위해서는 persistence.xml에서 다음을 수정해야 한다.

```
<property name="hibernate.hbm2ddl.auto" value="create"/>
```

```
1697
1698 4)다음으로 변경한다.
1699
1700 <property name="hibernate.hbm2ddl.auto" value="update"/>
1701
1702 5)이렇게 변경하면 새롭게 생성하지 않고 기존의 Table을 재사용한다.
1703 6)다음의 Data를 수행해서 3명의 User를 Insert한다.
1704
1705 chulsu, 34, 부산, 남, 김철수
1706 younghee, 44, 대전, 여, 이영희
1707
1708
1709 10. Data 검색
1710 1)JPAClient.java를 수정한다.
1711
1712 public static void main(String[] args) {
1713     //EntityManager 생성
1714     EntityManagerFactory emf =
1715         Persistence.createEntityManagerFactory("jpademo");
1716     EntityManager em = emf.createEntityManager();
1717
1718     try {
1719         // User 검색
1720
1721         User user = em.find(User.class, "jimin");
1722         System.out.println("jimin --> " + user);
1723     }catch(Exception e) {
1724         e.printStackTrace();
1725     }finally {
1726         em.close();
1727         emf.close();
1728     }
1729 }
1730
1731 2)실행
1732
1733 Hibernate:
1734 select
1735     user0_.userid as userid1_0_0_,
1736     user0_.age as age2_0_0_,
1737     user0_.city as city3_0_0_,
1738     user0_.gender as gender4_0_0_,
1739     user0_.username as username5_0_0_
1740 from
1741     User user0_
1742 where
1743     user0_.userid=?
1744 jimin --> User [userid=jimin, username=한지민, gender=여, age=24, city=서울]
1745
1746
1747 11. Entity 수정
1748 1)JPAClient.java 수정
```

```
1749
1750 public class JPAClient {
1751     public static void main(String[] args) {
1752         //EntityManager 생성
1753         EntityManagerFactory emf =
1754             Persistence.createEntityManagerFactory("jpademo");
1755         EntityManager em = emf.createEntityManager();
1756
1757         //Transaction 생성
1758         EntityTransaction tx = em.getTransaction();
1759
1760         try {
1761             //Transaction 시작
1762             tx.begin();
1763
1764             // 수정할 User 조회
1765             User user = em.find(User.class, "younghee");
1766             user.setCity("광주");
1767             user.setAge(55);
1768
1769             // Transaction commit
1770             tx.commit();
1771         } catch (Exception e) {
1772             e.printStackTrace();
1773             // Transaction rollback
1774             tx.rollback();
1775         } finally {
1776             em.close();
1777             emf.close();
1778         }
1779     }
1780 }
```

2) 실행

```
1784 Hibernate:
1785     select
1786         user0_.userid as userid1_0_0_,
1787         user0_.age as age2_0_0_,
1788         user0_.city as city3_0_0_,
1789         user0_.gender as gender4_0_0_,
1790         user0_.username as username5_0_0_
1791     from
1792         User user0_
1793     where
1794         user0_.userid=?
1795 Hibernate:
1796     update
1797         User
1798     set
1799         age=?,
1800         city=,
```

```
1801         gender=?,
1802         username=?
1803     where
1804         userid=?
1805
1806
1807 12. Entity 삭제
1808     1)JPAClient.java 수정
1809
1810         //Transaction 생성
1811         EntityTransaction tx = em.getTransaction();
1812
1813         try {
1814             //Transaction 시작
1815             tx.begin();
1816
1817             // 삭제할 User 조회
1818             User user = em.find(User.class, "younghee");
1819             em.remove(user);
1820
1821             // Transaction commit
1822             tx.commit();
1823         }catch(Exception e) {
```

```
1824
1825 2)실행
1826
1827     Hibernate:
1828         select
1829             user0_.userid as userid1_0_0_,
1830             user0_.age as age2_0_0_,
1831             user0_.city as city3_0_0_,
1832             user0_.gender as gender4_0_0_,
1833             user0_.username as username5_0_0_
1834         from
1835             User user0_
1836         where
1837             user0_.userid=?
1838     Hibernate:
1839         delete
1840         from
1841             User
1842         where
```

```
1843
1844
1845 13. 여러 Record 조회와 JPQL
1846     1)한 건의 Record 조회는 find()를 사용한다.
1847     2)하지만, 여러 건의 Record를 조회하기 위해서는 JPQL(Java Persistence Query Language)라는 JPA에서
        제공하는 별도의 Query 명령어를 사용해야 한다.
1848     3)JPAClient.java 수정
1849
1850         public static void main(String[] args) {
1851             //EntityManager 생성
```

```
1852     EntityManagerFactory emf =
1853         Persistence.createEntityManagerFactory("jpademo");
1854     EntityManager em = emf.createEntityManager();
1855
1856     //Transaction 생성
1857     EntityTransaction tx = em.getTransaction();
1858
1859     try {
1860         //Transaction 시작
1861         tx.begin();
1862
1863         User user = new User();
1864         user.setUserid("hojune");
1865         user.setUsername("이호준");
1866         user.setAge(30);
1867         user.setGender("남");
1868         user.setCity("수원");
1869
1870         // User 등록
1871         em.persist(user);
1872
1873         // Transaction commit
1874         tx.commit();
1875
1876         // 여러 Record 조회
1877         String jpql = "SELECT u FROM User u ORDER BY u.userid DESC";
1878         List<User> userList = em.createQuery(jpql, User.class).getResultList();
1879         for(User usr : userList) {
1880             System.out.println(usr);
1881         }
1882
1883     }catch(Exception e) {
1884         e.printStackTrace();
1885         // Transaction rollback
1886         tx.rollback();
1887     }finally {
1888         em.close();
1889         emf.close();
1890     }
1891 }
1892
1893 4)실행
1894
1895 Hibernate:
1896     insert
1897     into
1898         User
1899         (age, city, gender, username, userid)
1900     values
1901         (?, ?, ?, ?, ?)
1902
1903 Hibernate:
1904     select
```

```

1904         user0_.userid as userid1_0_,
1905         user0_.age as age2_0_,
1906         user0_.city as city3_0_,
1907         user0_.gender as gender4_0_,
1908         user0_.username as username5_0_
1909     from
1910         User user0_
1911     order by
1912         user0_.userid DESC
1913     User [userid=mija, username=이미자, gender=여, age=60, city=대구]
1914     User [userid=jimin, username=한지민, gender=여, age=24, city=서울]
1915     User [userid=hojune, username=이호준, gender=남, age=30, city=수원]
1916     User [userid=chulsu, username=김철수, gender=남, age=34, city=부산]

```

1917
1918
1919 -----

1920 Task8. 정적 Page 만들기

1921 1. Spring Boot project 생성

1922 1)Package Explorer > right-click > New > Spring Starter Project

1923 2)다음 각 항목의 값을 입력한 후, Next 클릭한다.

1924 -Service URL :<http://start.spring.io>

1925 -Name : springweb

1926 -Type : Maven

1927 -Packaging : jar

1928 -Java Version : 8

1929 -Language : Java

1930 -Group : com.example

1931 -Artifact : springweb

1932 -Version : 0.0.1-SNAPSHOT

1933 -Description : Demo project for Spring Boot

1934 -Package : com.example.biz

1935

1936 3)다음의 각 항목을 선택한 후 Finish 클릭

1937 -Spring Boot Version : 2.2.1

1938 -Select Spring Web, Spring Boot DevTools

1939

1940 2. Controller 생성

1941 1)src/main/java/com.example.biz > right-click > New > Class

1942 2)Name : HomeController

1943

1944 package com.example.biz;

1945

1946 import org.springframework.stereotype.Controller;

1947 import org.springframework.web.bind.annotation.GetMapping;

1948 import org.springframework.web.servlet.ModelAndView;

1949

1950 @Controller

1951 public class HomeController {

1952

1953 @GetMapping("/")

1954 public ModelAndView home(ModelAndView mav) {

1955 mav.setViewName("index.html");


```

1956         return mav;
1957     }
1958 }
1959
1960 3. static file 생성
1961 1)src/main/resources/static/images folder 생성
1962     -spring-boot.png 추가할 것
1963 2)src/main/resources/static/js folder 생성
1964     -jquery-3.4.1.min.js 추가할 것
1965 3)src/main/resources/static/css folder 생성
1966     -bootstrap.min.css 추가할 것
1967 4)src/main/resources/static/index.html
1968
1969 <!DOCTYPE html>
1970 <html>
1971 <head>
1972 <meta charset="UTF-8">
1973 <title>Home page</title>
1974 <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css" />
1975 <script src="js/jquery-3.4.1.min.js"> </script>
1976 <script>
1977     $(document).ready(function(){
1978         alert("Hello, Spring Boot World!!!");
1979     });
1980 </script>
1981 </head>
1982 <body>
1983 <div>
1984     
1985 </div>
1986 <div class="jumbotron">
1987     <h1>Hello, Spring Boot World</h1>
1988     <p>...</p>
1989     <p><a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a></p>
1990 </div>
1991 </body>
1992 </html>
1993
1994 4. Spring Boot에서는 template을 사용하지 않을 경우 기본적으로 static file은 src/main/resources/static에서
1995 찾는다.
1996 5. 이럴 때는 반드시 file의 확장자 .html까지 넣어야 한다.
1997 6. springweb Project > right-click > Run As > Spring Boot App
1998 7. http://localhost:8080/
1999
2000 -----
2001 Task9. JSP Page 만들기
2002 1. Spring Boot에서는 JSP 사용을 권장하지 않는다.
2003 2. 'jar' 형식으로 동작하지 않고 War file로 배포해야 하는 등의 몇 가지 제약이 있어서이기도 하지만, 가장 큰
2004 이유는 이미 JSP 자체가 Server 측 언어로 그 사용 빈도가 줄고 있기 때문이다.
2005 3. view 부분에 code가 섞여서 logic을 분리하기 어렵고, HTML과 같은 tag를 사용하므로 HTML 편집기 등에
2006 서 JSP 삽입 부분을 분리하기 어려우며 Visual 편집기 등에서도 사용이 어렵다.

```

2005 4. 그래서 template을 통해 code를 분리해야 할 필요가 있는 것이다.

2006 5. Spring Boot project 생성

2007 1)Package Explorer > right-click > New > Spring Starter Project

2008 2)다음 각 항목의 값을 입력 후 Next 클릭

2009 -Service URL :<http://start.spring.io>

2010 -Name : springjspdemo

2011 -Type : Maven

2012 -Packaging : jar

2013 -Java Version : 8

2014 -Language : Java

2015 -Group : com.example

2016 -Artifact : springjspdemo

2017 -Version : 0.0.1-SNAPSHOT

2018 -Description : Demo project for Spring Boot

2019 -Package : com.example.biz

2020 3)각 항목 선택 후 Finish 클릭

2021 -Spring Boot Version : 2.2.1

2022 -Select Spring Web > Finish

2023

2024 6. pom.xml

2025 1)jstl을 위해

2026 <dependency>

2027 <groupId>javax.servlet</groupId>

2028 <artifactId>jstl</artifactId>

2029 <version>1.2</version>

2030 </dependency>

2031

2032 2)JSP를 위해

2033 <dependency>

2034 <groupId>org.apache.tomcat</groupId>

2035 <artifactId>tomcat-jasper</artifactId>

2036 <version>9.0.27</version>

2037 </dependency>

2038

2039 3)pom.xml > right-click > Run As > Maven install

2040 [INFO] BUILD SUCCESS

2041

2042

2043 7. folder 준비

2044 1)src/main folder 안에 webapp folder 생성

2045 2)webapp folder 안에 WEB-INF folder 생성

2046 3)WEB-INF folder 안에 jsp folder 생성

2047

2048 8. 만일 JSP를 template으로 사용하는 경우에는 이 WEB-INF folder 안에 template file을 준비할 필요가 있다.

2049 9. src/main/resources/application.properties code 추가

2050 -application.properties > right-click > Open with > Generic Editor

2051

2052 spring.mvc.view.prefix : /WEB-INF/jsp/

2053 spring.mvc.view.suffix : .jsp

2054

2055 10. jsp folder 안에 jsp file 생성

2056 1)index.jsp

```
2057
2058 <%@ page language="java" contentType="text/html; charset=UTF-8"
2059     pageEncoding="UTF-8"%>
2060 <%@ page import="java.util.Date, java.text.SimpleDateFormat" %>
2061 <!DOCTYPE html>
2062 <html>
2063 <head>
2064 <meta charset="UTF-8">
2065 <title>Insert title here</title>
2066 </head>
2067 <body>
2068     <h1>Index page</h1>
2069     <%=new SimpleDateFormat("yyyy년 MM월 dd일").format(new Date()) %>
2070 </body>
2071 </html>
```

```
2072
2073 11. com.example.biz.HomeController.java
2074
```

```
2075     package com.example.biz;
2076
2077     import org.springframework.stereotype.Controller;
2078     import org.springframework.web.bind.annotation.GetMapping;
2079
2080     @Controller
2081     public class HomeController {
2082
2083         @GetMapping("/")
2084         public String index() {
2085             return "index";
2086         }
2087     }
```

```
2088
2089 12. 실행
```

```
2090 http://localhost:8080/
```

```
2091
2092 Index page
2093 2019년 11월 09일
2094
```

```
2095
2096 -----
```

```
2097 Task10. thymeleaf template 사용하기
```

```
2098 1. Spring Boot project 생성
```

```
2099 1)Package Explorer > right-click > New > Spring Starter Project
```

```
2100 2)다음의 각 항목 입력 후 Next 클릭
```

```
2101 -Service URL :http://start.spring.io
```

```
2102 -Name : MyBootWeb
```

```
2103 -Type : Maven
```

```
2104 -Packaging : jar
```

```
2105 -Java Version : 8
```

```
2106 -Language : Java
```

```
2107 -Group : com.example
```

```
2108 -Artifact : MyBootWeb
```

```
2109 -Version : 0.0.1-SNAPSHOT
2110 -Description : Demo project for Spring Boot
2111 -Package : com.example.biz
2112
2113 3)각 항목 선택 후 Finish 클릭
2114 -Spring Boot Version : 2.2.1
2115 -Select Spring Web > Finish
2116
2117 2. src/main/java/com.example.biz.MyBootWebApplication.java 수정하기
2118
2119 package com.example.biz;
2120
2121 import org.springframework.boot.SpringApplication;
2122 import org.springframework.boot.autoconfigure.SpringBootApplication;
2123
2124 @SpringBootApplication
2125 public class MyBootWebApplication {
2126
2127     public static void main(String[] args) {
2128         SpringApplication.run(MyBootWebApplication.class, args);
2129     }
2130 }
2131
2132 3. 실행
2133 1)MyBootWebApplication.java > right-click > Run As > Spring Boot App
2134 -http://localhost:8080/
2135
2136 Whitelabel Error Page
2137 This application has no explicit mapping for /error, so you are seeing this as a fallback.
2138
2139 Fri Nov 08 23:25:37 KST 2019
2140 There was an unexpected error (type=Not Found, status=404).
2141 No message available
2142
2143 4. Controller 작성하기
2144 1)com.example.biz > right-click > New > Class
2145 2)Name : HelloController > Finish
2146 3)HelloController.java
2147
2148 package com.example.biz;
2149
2150 import org.springframework.web.bind.annotation.RequestMapping;
2151 import org.springframework.web.bind.annotation.RestController;
2152
2153 @RestController
2154 public class HelloController {
2155
2156     @RequestMapping("/")
2157     public String index() {
2158         return "Hello Spring Boot World!";
2159     }
2160 }
```

```
2161
2162 5. project > right-click > Run As > Spring Boot App
2163 6. http://localhost:8080/
2164
2165     Hello Spring Boot World!
2166
2167 7. 매개변수 전달
2168     1)HelloController.java 수정
2169
2170         @RequestMapping("/{num}")
2171         public String index(@PathVariable int num) {
2172             int result = 0;
2173             for(int i = 1 ; i <= num ; i++) result += i;
2174             return "total : " + result;
2175         }
2176
2177 8. project > right-click > Run As > Spring Boot App
2178 9. http://localhost:8080/100
2179
2180     total : 5050
2181
2182 10. 객체를 JSON으로 출력하기
2183     1)src/main/java/com.example.biz/Student.java 생성
2184
2185     package com.example.biz;
2186
2187     public class Student {
2188         private int userid;
2189         private String name;
2190         private int age;
2191         private String address;
2192         public Student(int userid, String name, int age, String address) {
2193             this.userid = userid;
2194             this.name = name;
2195             this.age = age;
2196             this.address = address;
2197         }
2198         public int getUserid() {
2199             return userid;
2200         }
2201         public void setUserid(int userid) {
2202             this.userid = userid;
2203         }
2204         public String getName() {
2205             return name;
2206         }
2207         public void setName(String name) {
2208             this.name = name;
2209         }
2210         public int getAge() {
2211             return age;
2212         }
2212     }
```

```
2213     public void setAge(int age) {
2214         this.age = age;
2215     }
2216     public String getAddress() {
2217         return address;
2218     }
2219     public void setAddress(String address) {
2220         this.address = address;
2221     }
2222 }
2223
2224 2)HelloController.java 수정
2225
2226 @RestController
2227 public class HelloController {
2228     String [] names = {"조용필", "이미자", "설운도"};
2229     int [] ages = {56, 60, 70};
2230     String [] addresses = {"서울특별시", "부산광역시", "대전광역시"};
2231
2232     @RequestMapping("/{userid}")
2233     public Student index(@PathVariable int userid) {
2234         return new Student(userid, names[userid], ages[userid], addresses[userid]);
2235     }
2236 }
2237
2238 3)http://localhost:8080/1
2239
2240 {"userid":1,"name":"이미자","age":60,"address":"부산광역시"}
2241
2242
2243 11. thymeleaf 추가하기
2244 1)pom.xml 수정하기
2245     -Select Dependencies tab > Add
2246     -Group Id : org.springframework.boot
2247     -Artifact Id : spring-boot-starter-thymeleaf
2248     -Version :
2249     -Scope : compile
2250     -OK
2251
2252     <dependency>
2253         <groupId>org.springframework.boot</groupId>
2254         <artifactId>spring-boot-starter-thymeleaf</artifactId>
2255         <version>2.2.1.RELEASE</version>
2256     </dependency>
2257
2258 2)HelloController.java 수정
2259
2260     package com.example.biz;
2261
2262     import org.springframework.stereotype.Controller;
2263     import org.springframework.web.bind.annotation.RequestMapping;
2264
```

```

2265     @Controller
2266     public class HelloController {
2267
2268         @RequestMapping("/")
2269         public String index() {
2270             return "index";
2271         }
2272
2273 3)template file 생성
2274     -src/main/resources/templates > right-click > New > Other...> Web > HTML File > Next
2275     -File name : index.html
2276
2277     <!doctype html>
2278     <html lang="en">
2279         <head>
2280             <meta charset="UTF-8" /> <!--반드시 종결 tag 필요 -->
2281             <title>Index Page</title>
2282             <style type="text/css">
2283                 h1 { font-size:18pt; font-weight:bold; color:gray; }
2284                 body { font-size:13pt; color:gray; margin:5px 25px; }
2285             </style>
2286         </head>
2287         <body>
2288             <h1>Hello! Spring Boot with Thymeleaf</h1>
2289             <p>This is sample web page.</p>
2290         </body>
2291     </html>
2292
2293 4)<a href="http://localhost:8080/">http://localhost:8080/
2294
2295     Hello! Spring Boot with Thymeleaf
2296     This is sample web page.
2297
2298 5)template에 값 표시하기
2299     -index.html code 수정
2300
2301     <body>
2302         <h1>Hello! Spring Boot with Thymeleaf</h1>
2303         <p class="msg" th:text="${msg}"></p>
2304     </body>
2305
2306     -HelloController.java 수정
2307
2308     @Controller
2309     public class HelloController {
2310
2311         @RequestMapping("/{num}")
2312         public String index(@PathVariable int num, Model model) {
2313             int result = 0;
2314             for(int i = 1 ; i <= num ; i++) result += i;
2315             model.addAttribute("msg", "total : " + result);
2316             return "index";

```

```
2317     }
2318   }
2319
2320 6)http://localhost:8080/100
2321
2322   Hello! Spring Boot with Thymeleaf
2323   total : 5050
2324
2325
2326 7)ModelAndView class 사용하기
2327   -HelloController.java 수정
2328
2329   @Controller
2330   public class HelloController {
2331
2332       @RequestMapping("/{num}")
2333       public ModelAndView index(@PathVariable int num, ModelAndView mav) {
2334           int result = 0;
2335           for(int i = 1 ; i <= num ; i++) result += i;
2336           mav.addObject("msg", "total : " + result);
2337           mav.setViewName("index");
2338           return mav;
2339       }
2340   }
2341
2342 8)http://localhost:8080/100
2343
2344   Hello! Spring Boot with Thymeleaf
2345   total : 5050
2346
2347 9)form 사용하기
2348   -index.html 수정
2349
2350   <!doctype html>
2351   <html lang="en">
2352       <head>
2353           <meta charset="UTF-8" />
2354           <title>Index Page</title>
2355           <style type="text/css">
2356               h1 { font-size:18pt; font-weight:bold; color:gray; }
2357               body { font-size:13pt; color:gray; margin:5px 25px; }
2358           </style>
2359       </head>
2360       <body>
2361           <h1>Hello!</h1>
2362           <p th:text="{msg}"> ${msg}</p>
2363           <form method="post" action="/send">
2364               <input type="text" name="txtName" th:value="{value}" />
2365               <input type="submit" value="Send" />
2366           </form>
2367       </body>
2368   </html>
```



```
2369
2370 -HelloController.java 수정
2371
2372 @Controller
2373 public class HelloController {
2374
2375     @RequestMapping(value = "/", method = RequestMethod.GET)
2376     public ModelAndView index(ModelAndView mav) {
2377         mav.setViewName("index");
2378         mav.addObject("msg", "Please write your name...");
2379         return mav;
2380     }
2381
2382     @RequestMapping(value = "/send", method = RequestMethod.POST)
2383     public ModelAndView send(@RequestParam("txtName") String name,
2384                             ModelAndView mav) {
2385         mav.addObject("msg", "안녕하세요! " + name + "님!");
2386         mav.addObject("value", name);
2387         mav.setViewName("index");
2388         return mav;
2389     }
2390 }
2391
2392 10) http://localhost:8080
2393
2394 11)기타 form controller
2395 -index.html 수정
2396
2397 <!doctype html>
2398 <html lang="en">
2399 <head>
2400 <meta charset="UTF-8" />
2401 <title>Index Page</title>
2402 <style type="text/css">
2403 h1 {
2404     font-size: 18pt;
2405     font-weight: bold;
2406     color: gray;
2407 }
2408
2409 body {
2410     font-size: 13pt;
2411     color: gray;
2412     margin: 5px 25px;
2413 }
2414 </style>
2415 </head>
2416 <body>
2417     <h1>Hello!</h1>
2418     <pre th:text="{msg}">Please wait...</pre>
2419     <form method="post" action="/">
2420     <div>
```

```

2421         <input type="checkbox" id="ckeck1" name="check1" /> <label
2422             for="ckeck1">체크</label>
2423     </div>
2424     <div>
2425         <input type="radio" id="male" name="gender" value="male" /> <label
2426             for="male">남성</label>
2427     </div>
2428     <div>
2429         <input type="radio" id="female" name="gender" value="female" /> <label
2430             for="female">여성</label>
2431     </div>
2432     <div>
2433         <select name="selOs" size="4">
2434             <option>--선택--</option>
2435             <option value="Windows">windows</option>
2436             <option value="MacOS">MacOS</option>
2437             <option value="Linux">Linux</option>
2438         </select>
2439         <select name="selEditors" size="4" multiple="multiple">
2440             <option>--선택--</option>
2441             <option value="Notepad">Notepad</option>
2442             <option value="Editplus">Editplus</option>
2443             <option value="Visual Studio Code">Visual Studio Code</option>
2444             <option value="Sublime Text">Sublime Text</option>
2445             <option value="Eclipse">Eclipse</option>
2446         </select>
2447     </div>
2448     <input type="submit" value="전송" />
2449 </form>
2450 </body>
2451 </html>

```

2452 12)HelloController.java 수정

```

2453
2454
2455 @Controller
2456 public class HelloController {
2457
2458     @RequestMapping(value = "/", method = RequestMethod.GET)
2459     public ModelAndView index(ModelAndView mav) {
2460         mav.setViewName("index");
2461         mav.addObject("msg", "값을 입력후 전송버튼을 눌러주세요.");
2462         return mav;
2463     }
2464
2465     @RequestMapping(value = "/", method = RequestMethod.POST)
2466     public ModelAndView send(@RequestParam(value="check1", required=false) boolean check1,
2467                             @RequestParam(value="gender", required=false) String gender,
2468                             @RequestParam(value="selOs", required=false) String selOs,
2469                             @RequestParam(value="selEditors", required=false) String [] selEditors,
2470                             ModelAndView mav) {
2471
2472         String result = "";

```

```

2473         try {
2474             result = "check : " + check1 +
2475                 ", gender : " + gender +
2476                 ", OS : " + selOs +
2477                 "\nEditors : ";
2478         }catch(NullPointerException ex) {}
2479         try {
2480             result += selEditors[0];
2481             for(int i = 1 ; i < selEditors.length ; i++) result += ", " + selEditors[i];
2482         }catch(NullPointerException ex) {
2483             result += "null";
2484         }
2485         mav.addObject("msg", result);
2486         mav.setViewName("index");
2487         return mav;
2488     }
2489 }

```

2491 13) <http://localhost:8080>

2494 12. Redirect

2495 1) index.html 수정

```

2496
2497     <body>
2498         <h1>Hello! index.</h1>
2499     </body>

```

2501 2) HelloController.java 수정

```

2502
2503     @Controller
2504     public class HelloController {
2505
2506         @RequestMapping("/")
2507         public ModelAndView index(ModelAndView mav) {
2508             mav.setViewName("index");
2509             return mav;
2510         }
2511
2512         @RequestMapping("/other")
2513         public String other() {
2514             return "redirect:/";
2515         }
2516
2517         @RequestMapping("/home")
2518         public String home() {
2519             return "forward:/";
2520         }
2521     }

```

2523 3) redirect와 forward 차이점 구분하기

```
2525
2526 -----
2527 Task11. thymeleaf template 사용하기2
2528 1. Spring Boot project 생성
2529     1)Package Explorer > right-click > New > Spring Starter Project
2530     2)다음 각 항목의 값을 입력 후 Next 클릭
2531         -Service URL :http://start.spring.io
2532         -Name : templatedemo
2533         -Type : Maven
2534         -Packaging : Jar
2535         -Java Version : 8
2536         -Language : Java
2537         -Group : com.example
2538         -Artifact : templatedemo
2539         -Version : 0.0.1-SNAPSHOT
2540         -Description : Demo project for Spring Boot
2541         -Package : com.example.biz
2542     3)다음 각 항목 선택 후 Finish 클릭
2543         -Spring Boot Version : 2.2.1
2544         -Select Spring Web > Web, Template Engines > Thymeleaf > Finish
2545
2546
2547 2. HomeController.java 생성
2548     1)com.example.biz > right-click > New > Class
2549     2)Name : HomeController
2550
2551     package com.example.biz;
2552
2553     import org.springframework.stereotype.Controller;
2554     import org.springframework.web.bind.annotation.GetMapping;
2555
2556     @Controller
2557     public class HomeController {
2558
2559         @GetMapping("/")
2560         public String home() {
2561             return "index";
2562         }
2563     }
2564
2565 3)index.html 생성
2566     -src/main/resources/templates > New > Other > Web > HTML File > Next
2567     -File name : index.html
2568
2569     <!DOCTYPE html>
2570     <html>
2571     <head>
2572     <meta charset="UTF-8" />
2573     <title>Insert title here</title>
2574     </head>
2575     <body>
2576     <h1>Hello Page</h1>
```

```
2577         <p th:text="${new java.util.Date().toString()}"> </p>
2578     </body>
2579 </html>
2580
2581 -실행
2582 --http://localhost:8080
2583
2584     Hello Page
2585     Sat Nov 09 00:02:32 KST 2019
2586
2587 3. Utility Object 사용하기
2588     1)index.html 수정
2589
2590     <body>
2591         <h1>Hello Page</h1>
2592         <p th:text="${#dates.format(new java.util.Date(), 'dd/MM/yyyy HH:mm')}"> </p>
2593         <p th:text="${#numbers.formatInteger(1234,7)}" > </p>
2594         <p th:text="${#strings.toUpperCase('Welcome to Spring.')}"> </p>
2595     </body>
2596
2597     2)실행
2598
2599     Hello Page
2600     09/11/2019 00:15
2601
2602     0001234
2603
2604     WELCOME TO SPRING.
2605
2606
2607 4. 매개변수에 접근하기
2608     1)index.html의 <body> 부분을 아래와 같이 수정한다.
2609
2610     <body>
2611         <h1>Helo page</h1>
2612         <p th:text="'from parameter... id=' + ${param.id[0]} + ',name=' + param.name[0]'"> </p>
2613     </body>
2614
2615     2)HomeController.java를 수정한다.
2616
2617     @RequestMapping("/")
2618     public ModelAndView index(ModelAndView mav) {
2619         mav.setViewName("index");
2620         return mav;
2621     }
2622
2623     @RequestMapping("/home")
2624     public String home() {
2625         return "forward:/";
2626     }
2627
2628     3)그리고 id와 name을 query string으로 지정해서 접속한다.
```

2629 -<http://localhost:8080/home/?id=javaexpert&name=Springboot>
2630 -결과는 아래와 같다.
2631
2632 Helo page
2633 from parameter... id=javaexpert,name=Springboot
2634
2635 4)controller를 거치지 않고 template내에서 직접 전달된 값을 사용할 수 있다.
2636 5)중요한 것은 param내의 id나 name 배열에서 첫 번째 요소를 지정해서 추출한다는 것이다.
2637 6)그 이유는 query string으로 값을 전송할 때 같은 이름의 값을 여러 개 전송하기 위해서다.
2638 7)예를 들면, <http://localhost:8080/home/?id=123&id=456&name=javaexpert&name=peter>
2639 8)이렇게 하면 param에서 추출하는 id와 name이 각각 {123,456}, {javaexpert,peter}가 되는 것이기 때문이다.
2640
2641 9)여기서 사용하고 있는 th:text의 값을 보면 큰따옴표 안에 다시 작은 따옴표를 사용해서 값을 작성하고
있다.
2642 10)이것은 OGNL로 text literal을 작성할 때 사용하는 방식이다.
2643 11)이렇게 하면 다수의 literal을 연결할 때 큰 따옴표안에 작은 따옴표 literal을 사용할 수 있게 된다.
2644
2645 th:text="one two three"
2646 th:text="'one' + ' two ' + 'three'"
2647
2648
2649 5. Message식 사용하기
2650 1)src/main/resources > right-click > New > File
2651 2)File name : messages.properties > Finish
2652
2653 content.title=Message sample page.
2654 content.message=This is sample message from properties.
2655
2656 3)index.html 수정
2657
2658 <body>
2659 <h1 th:text="#{content.title}">Hello page</h1>
2660 <p th:text="#{content.message}"></p>
2661 </body>
2662
2663 4)실행
2664
2665 Message sample page.
2666 This is sample message from properties.
2667
2668
2669 6. Link식과 href
2670 1)index.html
2671
2672 <body>
2673 <h1 th:text="#{content.title}">Helo page</h1>
2674 <p><a th:href="@{/home/{orderId}(orderId=\${param.id[0]})}">link</p>
2675 </body>
2676
2677 2)접속할 때 query string에 id를 지정한다.
2678 3)예를 들어 <http://localhost:8080/?id=123>에 접속하면 link에는 /home/123이 설정된다.

```
2679
2680
2681 7. 선택 객체와 변수식
2682 1)src/main/java/com.example.biz > right-click > New > Class
2683 2)Name : Member
2684
2685     package com.example.biz;
2686
2687     public class Member {
2688         private int id;
2689         private String username;
2690         private int age;
2691
2692         public Member(int id, String username, int age) {
2693             this.id = id;
2694             this.username = username;
2695             this.age = age;
2696         }
2697
2698         public int getId() {
2699             return id;
2700         }
2701         public void setId(int id) {
2702             this.id = id;
2703         }
2704         public String getUsername() {
2705             return username;
2706         }
2707         public void setUsername(String username) {
2708             this.username = username;
2709         }
2710         public int getAge() {
2711             return age;
2712         }
2713         public void setAge(int age) {
2714             this.age = age;
2715         }
2716     }
2717
2718 3)HomeController.java 수정
2719
2720     @RequestMapping("/")
2721     public ModelAndView index(ModelAndView mav) {
2722         mav.setViewName("index");
2723         mav.addObject("msg","Current data.");
2724         Member obj = new Member(123, "javaexpert",24);
2725         mav.addObject("object",obj);
2726         return mav;
2727     }
2728
2729 4)index.html 수정
2730
```

```

2731 <!DOCTYPE HTML>
2732 <html xmlns:th="http://www.thymeleaf.org">
2733 <head>
2734   <title>top page</title>
2735   <meta charset="UTF-8" />
2736   <style>
2737     h1 { font-size:18pt; font-weight:bold; color:gray; }
2738     body { font-size:13pt; color:gray; margin:5px 25px; }
2739     tr { margin:5px; }
2740     th { padding:5px; color:white; background:darkgray; }
2741     td { padding:5px; color:black; background:#e0e0ff; }
2742   </style>
2743 </head>
2744 <body>
2745   <h1 th:text="#{content.title}">Hello page</h1>
2746   <p th:text="${msg}">message.</p>
2747   <table th:object="${object}">
2748     <tr><th>ID</th><td th:text="*{id}"></td></tr>
2749     <tr><th>NAME</th><td th:text="*{username}"></td></tr>
2750     <tr><th>AGE</th><td th:text="*{age}"></td></tr>
2751   </table>
2752 </body>
2753 </html>

```

2754
2755 5)실행

2756 6)controller에서 object를 저장해둔 Member 값이 표 형태로 출력된다.

```

2757
2758
2759 -----
2760 Task12. Spring Boot와 JDBC 연동하기
2761 1. Spring Boot project 생성
2762   1)Package Explorer > right-click > New > Spring Starter Project
2763   2)다음의 각 항목의 값을 입력후 Next 클릭
2764     -Service URL :http://start.spring.io
2765     -Name : BootJdbcDemo
2766     -Type : Maven
2767     -Packaging : Jar
2768     -Java Version : 8
2769     -Language : Java
2770     -Group : com.example
2771     -Artifact : BootJdbcDemo
2772     -Version : 0.0.1-SNAPSHOT
2773     -Description : Demo project for Spring Boot
2774     -Package : com.example.biz
2775   3)다음 각 항목을 선택후 Finish 클릭
2776     -Spring Boot Version : 2.2.1
2777     -Select SQL > check MySQL, JDBC API
2778     -Select Spring Web > Finish
2779
2780   4)위에서 type을 선택시 고려사항
2781     -만일 Embedded된 tomcat으로 Stand-Alone형태로 구동시키기 위한 목적이라면 Packaging Type을 jar
       로 선택한다.

```


2782 -war로 선택할 경우, 기존과 같이 외부의 tomcat으로 deploy 하는 구조로 만들어진다.
2783 -물론, source의 최종배포의 형태가 server의 tomcat에 deploy해야 하는 구조라면 처음부터 war로 만
들어서 작업해도 상관없다.
2784 -jar로 선택하고, local에서 개발 및 test를 하다가 나중에 배포할 경우 war로 변경해서 배포를 할 수도
있다.

2785

2786 2. pom.xml

2787 1)jstl 사용을 위한

2788

2789 <dependency>

2790 <groupId>javax.servlet</groupId>

2791 <artifactId>jstl</artifactId>

2792 <version>1.2</version>

2793 </dependency>

2794

2795 2)jasper 사용을 위한

2796

2797 <dependency>

2798 <groupId>org.apache.tomcat</groupId>

2799 <artifactId>tomcat-jasper</artifactId>

2800 <version>9.0.27</version>

2801 </dependency>

2802

2803 3)MariaDB 사용을 위한

2804 <dependency>

2805 <groupId>org.mariadb.jdbc</groupId>

2806 <artifactId>mariadb-java-client</artifactId>

2807 <version>2.5.1</version>

2808 </dependency>

2809

2810

2811 3. src/main/resources/application.properties

2812

2813 spring.application.name=BootJDBCDemo

2814 spring.datasource.url=jdbc:mariadb://localhost:3306/test

2815 spring.datasource.driver-class-name=org.mariadb.jdbc.Driver

2816 spring.datasource.username=root

2817 spring.datasource.password=javamariadb

2818

2819 4. Table 생성

2820

2821 CREATE TABLE test.User

2822 (

2823 username VARCHAR(20) PRIMARY KEY,

2824 age TINYINT NOT NULL

2825);

2826

2827

2828 5. VO object 생성

2829 1)src/main/java > right-click > New > Package

2830 2)Name : com.example.vo

2831 3)com.example.vo > right-click > New > Class

```
2832 4)Name : UserVO
2833
2834 package com.example.vo;
2835
2836 public class UserVO {
2837     private String username;
2838     private int age;
2839
2840     public String getUsername() {
2841         return username;
2842     }
2843     public void setUsername(String username) {
2844         this.username = username;
2845     }
2846     public int getAge() {
2847         return age;
2848     }
2849     public void setAge(int age) {
2850         this.age = age;
2851     }
2852     @Override
2853     public String toString() {
2854         return "UserVO [username=" + username + ", age=" + age + "]";
2855     }
2856 }
```

2859 6. Dao object 생성

```
2860 1)src/main/java > right-click > New > Package
2861 2)Name : com.example.dao
2862 3)com.example.dao > right-click > New > Interface
2863 4)Name : UserDao
```

```
2864
2865 package com.example.dao;
2866
2867 import java.util.List;
2868
2869 import com.example.vo.UserVO;
2870
2871 public interface UserDao {
2872     int create(UserVO userVO);
2873     List<UserVO> readAll();
2874     UserVO read(String username);
2875     int update(UserVO userVO);
2876     int delete(String username);
2877 }
```

```
2879 5)com.example.dao > right-click > New > Class
```

```
2880 6)Name : UserDaoImpl
```

```
2881
2882 package com.example.dao;
2883
```

```
2884 import static org.mockito.Mockito.RETURNS_DEEP_STUBS;
2885
2886 import java.sql.ResultSet;
2887 import java.sql.SQLException;
2888 import java.util.List;
2889
2890 import org.springframework.beans.factory.annotation.Autowired;
2891 import org.springframework.jdbc.core.JdbcTemplate;
2892 import org.springframework.jdbc.core.RowMapper;
2893 import org.springframework.stereotype.Repository;
2894
2895 import com.example.vo.UserVO;
2896
2897 @Repository
2898 public class UserDaoImpl implements UserDao {
2899     @Autowired
2900     private JdbcTemplate jdbcTemplate;
2901
2902     class UserMapper implements RowMapper<UserVO> {
2903         public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2904             UserVO user = new UserVO();
2905             user.setUsername(rs.getString("username"));
2906             user.setAge(rs.getInt("age"));
2907             return user;
2908         }
2909     }
2910
2911     @Override
2912     public int create(UserVO userVO) {
2913         String sql = "INSERT INTO User(username, age) VALUES(?, ?)";
2914         return this.jdbcTemplate.update(sql, userVO.getUsername(), userVO.getAge());
2915     }
2916
2917     @Override
2918     public List<UserVO> readAll() {
2919         String sql = "SELECT * FROM User";
2920         return this.jdbcTemplate.query(sql, new UserMapper());
2921     }
2922
2923     @Override
2924     public UserVO read(String username) {
2925         String sql = "SELECT * FROM User WHERE username = ?";
2926         return this.jdbcTemplate.queryForObject(sql, new Object[] {username}, new UserMapper());
2927     }
2928
2929     @Override
2930     public int update(UserVO userVO) {
2931         String sql = "UPDATE User SET age = ? WHERE username = ?";
2932         return this.jdbcTemplate.update(sql, userVO.getAge(), userVO.getUsername());
2933     }
2934
2935     @Override
```

```
2936     public int delete(String username) {
2937         String sql = "DELETE FROM User WHERE username = ?";
2938         return this.jdbcTemplate.update(sql, username);
2939     }
2940
2941 }
2942
2943 7. Controller
2944 1)com.example.biz > right-click > New > Class
2945 2)Name : MainController
2946
2947     package com.example.biz;
2948
2949     import org.springframework.beans.factory.annotation.Autowired;
2950     import org.springframework.stereotype.Controller;
2951     import org.springframework.ui.Model;
2952     import org.springframework.web.bind.annotation.GetMapping;
2953     import org.springframework.web.bind.annotation.PostMapping;
2954
2955     import com.example.dao.UserDao;
2956     import com.example.vo.UserVO;
2957
2958     @Controller
2959     public class MainController {
2960         @Autowired
2961         private UserDao userDao;
2962
2963         @GetMapping("/")
2964         public String index() {
2965             return "index";
2966         }
2967
2968         @PostMapping("/user")
2969         public String insert(UserVO userVO) {
2970             this.userDao.create(userVO);
2971             return "redirect:/user";
2972         }
2973
2974         @GetMapping("/user")
2975         public String list(Model model) {
2976             model.addAttribute("users", this.userDao.readAll());
2977             return "list";
2978         }
2979     }
2980
2981
2982 8. static resources 준비
2983 1)src/main/resources/static/images folder 생성
2984   -spring-boot.png 추가할 것
2985 2)src/main/resources/static/js folder 생성
2986   -jquery-3.4.1.min.js 추가할 것
2987 3)src/main/resources/static/css folder 생성
```

```
2988 -style.css
2989
2990 @charset "UTF-8";
2991 body {
2992     background-color:yellow;
2993 }
2994 h1{
2995     color : blue;
2996 }
2997
2998 9. JSP를 위한 folder 준비
2999 1)기본적으로 Spring Boot 에서는 jsp파일을 인식이 되지 않는다.
3000 2)그래서 만일 jar로 packaging 한다면 embedded tomcat이 인식하는 web루트를 생성한다.
3001 3)src > main 폴더 밑에 webapp과 jsp가 위치할 폴더를 만들어준다.
3002 4)src/main folder 안에 webapp folder 생성
3003 5)webapp folder 안에 WEB-INF folder 생성
3004 6)WEB-INF folder 안에 jsp folder 생성
3005
3006 7)만일 JSP를 template으로 사용하는 경우에는 이 WEB-INF folder 안에 template file을 준비할 필요가 있
3007 다.
3008 8)folder를 다 만들었으면, WEB 루트(Context Root)로 인식할 있도록 환경설정을 아래와 같이 추가한다.
3009 -src/main/resources/application.properties code 추가
3010
3011 spring.mvc.view.prefix : /WEB-INF/jsp/
3012 spring.mvc.view.suffix : .jsp
3013
3014 10. jsp folder 안에 jsp file 생성
3015 1)index.jsp
3016
3017 <%@ page language="java" contentType="text/html; charset=UTF-8"
3018     pageEncoding="UTF-8"%>
3019 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3020 <!DOCTYPE html>
3021 <html>
3022 <head>
3023 <meta charset="UTF-8">
3024 <title>New User Insertion Page</title>
3025 <link rel="stylesheet" type="text/css" href="/css/style.css">
3026 <c:url var="jsurl" value="/js/jquery-3.3.1.slim.min.js" />
3027 <script src="${jsurl}"> </script>
3028 <script>
3029     $(document).ready(function() {
3030         alert("Hello, Spring Boot!");
3031     });
3032 </script>
3033 </head>
3034 <body>
3035
3036     
3037     <h1>New User Insertion Page</h1>
3038     <form action="/user" method="post">
3039         Name : <input type="text" name="username" /> <br />
```

```

3039         Age : <input type="number" name="age" /> <br />
3040         <button type="submit">Submit</button>
3041     </form>
3042 </body>
3043 </html>
3044
3045 2)list.jsp
3046
3047 <%@ page language="java" contentType="text/html; charset=UTF-8"
3048     pageEncoding="UTF-8"%>
3049 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3050 <!DOCTYPE html>
3051 <html>
3052 <head>
3053 <meta charset="UTF-8">
3054 <title>User List</title>
3055 <link rel="stylesheet" type="text/css" href="/css/style.css">
3056 </head>
3057 <body>
3058     <h1>User List</h1>
3059     <ul>
3060         <c:forEach var="user" items="${users}">
3061             <li>${user.username}(${user.age})</li>
3062         </c:forEach>
3063     </ul>
3064 </body>
3065 </html>
3066
3067

```

3068 11. src/main/java/com.example.biz/BootJdbcDemoApplication.java

```

3069
3070 package com.example.biz;
3071
3072 import org.springframework.boot.SpringApplication;
3073 import org.springframework.boot.autoconfigure.SpringBootApplication;
3074 import org.springframework.context.annotation.ComponentScan;
3075
3076 @SpringBootApplication
3077 @ComponentScan("com.example")    <-- 추가 code
3078 public class BootJdbcDemoApplication {
3079
3080     public static void main(String[] args) {
3081         SpringApplication.run(BootJdbcDemoApplication.class, args);
3082     }
3083 }
3084
3085 -@SpringBootApplication 은 다음의 annotation 3개를 모두 담고 있다.
3086     --@SpringBootConfiguration
3087     --@EnableAutoConfiguration
3088     --@ComponentScan

```

3089 -만약, AutoScan이 되어야 하는 component class들 - 대표적으로 @Controller, @Service, @Repository, @Component등-의 위치가 main class가 위치한 package보다 상위 package에 있거나, 하위가 아닌 다른

package에 있는 경우, scan이 되지 않는다.

3090 -Controller class가 main의 하위 class에 있으면 상관없지만, 예를 들어, main class가 다른 package나 하
 위 package가 아니면 아래와 같이 해줘야 한다.

3091 -명시적으로 ComponentScan을 할 Base Package를 지정해주면 된다.

3092 --ex) @ComponentScan(basePackages = "com.springboot.demo")

3093

3094 14)project > right-click > Run As > Spring Boot App

3095 15)<http://localhost:8080>

3096

3097

3098 -----

3099 Task13. Spring Boot와 JPA 연동하기

3100 1. Spring Boot의 Database 처리는 기본적으로 JPA 기술을 기반으로 하고 있다.

3101 2. 이 JPA를 Spring Framework에서 사용할 수 있게 한 것이 'Spring Data JPA' framework이다.

3102 3. Spring Boot에서는 JTA(Java Transaction API; Java EE에 transaction 처리를 제공), Spring ORM, Spring
 Aspects/Spring AOP는 'Spring Boot Starter Data JPA'라는 library를 사용해서 통합적으로 사용할 수 있다.

3103 4. 즉, 이 library는 각종 library를 조합해서 간단히 database 접속을 구현하게 한 기능이다.

3104 5. Spring Boot project 생성

3105 1)Package Explorer > right-click > New > Spring Starter Project

3106 2)다음 각 항목의 값을 입력한 후 Next 클릭

3107 -Service URL :<http://start.spring.io>

3108 -Name : JpaDemo

3109 -Type : Maven

3110 -Packaging : Jar

3111 -Java Version : 8

3112 -Language : Java

3113 -Group : com.example

3114 -Artifact : JpaDemo

3115 -Version : 0.0.1-SNAPSHOT

3116 -Description : Demo project for Spring Boot

3117 -Package : com.example.biz

3118 3)다음 각 항목을 선택한 후 Finish 클릭

3119 -Spring Boot Version : 2.2.1

3120 -Check Spring Data JPA, H2 Database Finish

3121

3122 6. Entity

3123 1)JPA에서 Entity라는 것은 Database에 저장하기 위해서 정의한 class이다.

3124 2)일반적으로 RDBMS에서 Table 같은 것이다.

3125 3)com.example.biz > right-click > New > Class

3126 4)Name : MemberVO

3127

3128 package com.example.biz;

3129

3130 import javax.persistence.Column;

3131 import javax.persistence.Entity;

3132 import javax.persistence.GeneratedValue;

3133 import javax.persistence.GenerationType;

3134 import javax.persistence.Id;

3135

3136 @Entity(name="Member")

3137 public class MemberVO {

3138 @Id

```
3139     @GeneratedValue(strategy = GenerationType.AUTO)
3140     private long id;
3141
3142     @Column
3143     private String username;
3144
3145     @Column
3146     private int age;
3147
3148     public MemberVO() {}
3149     public MemberVO(String username, int age) {
3150         this.username = username;
3151         this.age = age;
3152     }
3153
3154     public long getId() {
3155         return id;
3156     }
3157     public void setId(long id) {
3158         this.id = id;
3159     }
3160     public String getUsername() {
3161         return username;
3162     }
3163     public void setUsername(String username) {
3164         this.username = username;
3165     }
3166     public int getAge() {
3167         return age;
3168     }
3169     public void setAge(int age) {
3170         this.age = age;
3171     }
3172
3173     @Override
3174     public String toString() {
3175         return "MemberVO [id=" + id + ", username=" + username + ", age=" + age + "];"
3176     }
3177 }
```

5)여기서 주의할 점은 기본 생성자는 반드시 넣어야 한다.

3182 7. Repository

- 3183 1)Entity class를 구성했다면 이번엔 Repository interface를 만들어야 한다.
- 3184 2)Spring Framework에서는 Entity의 기본적인 삽입, 조회, 수정, 삭제가 가능하도록 CrudRepository라는 interface가 있다.
- 3185 3)com.example.biz > right-click > New > Interface
- 3186 4)Name : MemberRepository

```
3188     package com.example.biz;
```



```

3190 import java.util.List;
3191
3192 import org.springframework.data.jpa.repository.Query;
3193 import org.springframework.data.repository.CrudRepository;
3194 import org.springframework.data.repository.query.Param;
3195
3196 public interface MemberRepository extends CrudRepository<MemberVO, Long> {
3197     List<MemberVO> findByUsernameAndAgeLessThan(String username, int age);
3198
3199     @Query("select t from Member t where username= :username and age < :age")
3200     List<MemberVO> findByUsernameAndAgeLessThanSQL(@Param("username") String username,
3201         @Param("age") int age);
3202
3203     List<MemberVO> findByUsernameAndAgeLessThanOrderByAgeDesc(String username, int age);
3204 }

```

3205 -위의 코드는 실제로 MemberVO Entity를 이용하기 위한 Repository class이다.

3206 -기본적인 method 외에도 추가적인 method를 지정할 수 있다.

3207 -method 이름을 기반(Query Method)으로 해서 만들어도 되고 @Query를 이용해 기존의 SQL처럼 만들어도 된다.

3208 -findByUsernameAndAgeLessThan method와 findByUsernameAndAgeLessThanSQL method는 같은 결과를 출력하지만 전자의 method는 method 이름을 기반으로 한 것이고 후자의 method는 @Query annotation을 기반으로 해서 만든 것이다.

3209 -method 이름 기반으로 해서 만들면 추후에 사용할 때 method 이름만으로도 어떤 query인지 알 수 있다는 장점이 있다.

3210 -반대로 @Query annotation으로 만든 method는 기존의 source를 converting하는 경우 유용하게 사용할 수 있다.

3211 -@Query

3212 --다만 @Query annotation으로 query를 만들 때에 from 절에 들어가는 table은 Entity로 지정된 class 이름이다.

3213 -method 이름 기반 작성법

3214 -해당 부분은 Spring 문서
<https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.query-creation>를 통해 확인할 수 있다.

3215

3216

3217 8. Application 작성

3218 1)Entity 및 Repository가 준비 되었다면 실제로 @SpringBootApplication annotation이 있는 class에서 실제로 사용해 보자.

```

3219
3220 package com.example.biz;
3221
3222 import java.util.List;
3223
3224 import org.springframework.beans.factory.annotation.Autowired;
3225 import org.springframework.boot.CommandLineRunner;
3226 import org.springframework.boot.SpringApplication;
3227 import org.springframework.boot.autoconfigure.SpringBootApplication;
3228
3229 @SpringBootApplication
3230 public class JpaDemoApplication implements CommandLineRunner {
3231

```

```
3232     @Autowired
3233     MemberRepository memberRepository;
3234
3235     public static void main(String[] args) {
3236         SpringApplication.run(JpaDemoApplication.class, args);
3237     }
3238
3239     @Override
3240     public void run(String... args) throws Exception {
3241         memberRepository.save(new MemberVO("a", 10));
3242         memberRepository.save(new MemberVO("b", 15));
3243         memberRepository.save(new MemberVO("c", 10));
3244         memberRepository.save(new MemberVO("a", 5));
3245
3246         Iterable<MemberVO> list1 = memberRepository.findAll();
3247
3248         System.out.println("findAll() Method.");
3249         for( MemberVO m : list1){
3250             System.out.println(m.toString());
3251         }
3252
3253         System.out.println("findByUserNameAndAgeLessThan() Method.");
3254         List<MemberVO> list2 = memberRepository.findByUsernameAndAgeLessThan("a", 10);
3255         for( MemberVO m : list2){
3256             System.out.println(m.toString());
3257         }
3258
3259         System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
3260         List<MemberVO> list3 = memberRepository.findByUsernameAndAgeLessThanSQL("a", 10);
3261         for( MemberVO m : list3){
3262             System.out.println(m.toString());
3263         }
3264
3265         System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
3266         List<MemberVO> list4 =
3267             memberRepository.findByUsernameAndAgeLessThanOrderByAgeDesc("a", 15);
3268         for( MemberVO m : list4){
3269             System.out.println(m.toString());
3270         }
3271         memberRepository.deleteAll();
3272     }
3273 }
3274
3275
3276 9. 실행
3277
3278 findAll() Method.
3279 Member [id=1, name=a, age=10]
3280 Member [id=2, name=b, age=15]
3281 Member [id=3, name=c, age=10]
3282 Member [id=4, name=a, age=5]
```

```
3283 findByNameAndAgeLessThan() Method.
3284 Member [id=4, name=a, age=5]
3285 findByNameAndAgeLessThanSQL() Method.
3286 Member [id=4, name=a, age=5]
3287 findByNameAndAgeLessThanSQL() Method.
3288 Member [id=1, name=a, age=10]
3289 Member [id=4, name=a, age=5]
3290
3291
3292 -----
3293 Task14. Spring Boot JPA
3294 1. Spring Boot project 생성
3295 1)Package Explorer > right-click > New > Spring Starter Project
3296 2)다음의 각 항목의 값을 입력 후 Next 클릭
3297 -Service URL :http://start.spring.io
3298 -Name : BootJpaDemo
3299 -Type : Maven
3300 -Packaging : Jar
3301 -Java Version : 8
3302 -Language : Java
3303 -Group : com.example
3304 -Artifact : BootJpaDemo
3305 -Version : 0.0.1-SNAPSHOT
3306 -Description : Demo project for Spring Boot
3307 -Package : com.example.biz
3308 3)각 항목을 선택 후 Finish 클릭
3309 -Spring Boot Version : 2.2.1
3310 -Select Spring Boot DevTools, H2 Database, Spring Data JPA
3311 -Select Spring Web, Template Engines > Thymeleaf > Finish
3312
3313 4)DevTools
3314 -It provides developer tools.
3315 -These tools are helpful in application development mode.
3316 -One of the features of developer tool is automatic restart of the server for any change in code.
3317
3318
3319 2. Entity 작성하기
3320 1)com.example.biz > right-click > New > Package
3321 2)Name : com.example.biz.vo
3322 3)com.example.biz.vo > right-click > New > Class
3323 4)Name : User
3324
3325 package com.example.biz.vo;
3326
3327 import javax.persistence.Column;
3328 import javax.persistence.Entity;
3329 import javax.persistence.GeneratedValue;
3330 import javax.persistence.GenerationType;
3331 import javax.persistence.Id;
3332 import javax.persistence.Table;
3333
3334 @Entity
```

```
3335     @Table
3336     public class User {
3337
3338         @Id
3339         @GeneratedValue(strategy = GenerationType.AUTO)
3340         @Column
3341         private long id;
3342
3343         @Column(length = 20, nullable = false)
3344         private String username;
3345
3346         @Column(length = 100, nullable = true)
3347         private String email;
3348
3349         @Column(nullable = true)
3350         private Integer age;
3351
3352         @Column(nullable = true)
3353         private String memo;
3354
3355         public long getId() {
3356             return id;
3357         }
3358         public void setId(long id) {
3359             this.id = id;
3360         }
3361         public String getUsername() {
3362             return username;
3363         }
3364         public void setUsername(String username) {
3365             this.username = username;
3366         }
3367         public String getEmail() {
3368             return email;
3369         }
3370         public void setEmail(String email) {
3371             this.email = email;
3372         }
3373         public Integer getAge() {
3374             return age;
3375         }
3376         public void setAge(Integer age) {
3377             this.age = age;
3378         }
3379         public String getMemo() {
3380             return memo;
3381         }
3382         public void setMemo(String memo) {
3383             this.memo = memo;
3384         }
3385     }
3386
```

```
3387
3388 3. Repository 생성하기
3389 1)com.example.biz > right-click > New > Package
3390 2)Name : com.example.biz.dao
3391 3)com.example.biz.dao > right-click > New > Interface
3392 4)Name : UserRepository > Finish
3393
3394 package com.example.biz.dao;
3395
3396 import org.springframework.data.jpa.repository.JpaRepository;
3397 import org.springframework.stereotype.Repository;
3398
3399 import com.example.biz.vo.User;
3400
3401 @Repository("repository")
3402 public interface UserRepository extends JpaRepository<User, Long>{
3403
3404 }
3405 -JpaRepository라는 interface는 새로운 repository를 생성하기 위한 토대가 된다.
3406 -모든 Repository는 이 JpaRepository를 상속해서 작성한다.
3407
3408
3409 4. HelloController 작성
3410 1)com.example.biz > right-click > New > Class
3411 2)Name : HelloController
3412
3413 package com.example.biz;
3414
3415 import org.springframework.beans.factory.annotation.Autowired;
3416 import org.springframework.stereotype.Controller;
3417 import org.springframework.web.bind.annotation.RequestMapping;
3418 import org.springframework.web.servlet.ModelAndView;
3419
3420 import com.example.biz.dao.UserRepository;
3421 import com.example.biz.vo.User;
3422
3423 @Controller
3424 public class HelloController {
3425
3426     @Autowired
3427     UserRepository repository;
3428
3429     @RequestMapping("/")
3430     public ModelAndView index(ModelAndView mav) {
3431         mav.setViewName("index");
3432         mav.addObject("msg","this is sample content.");
3433         Iterable<User> list = repository.findAll();
3434         mav.addObject("data",list);
3435         return mav;
3436     }
3437 }
3438
```

3439 -UserRepository에는 findAll 같은 method가 정의되어 있지 않다.
3440 -이것은 부모 interface인 JpaRepository가 가지고 있는 method이다.
3441 -이를 통해 모든 entity가 자동으로 추출되는 것이다.
3442
3443
3444 5. JUnit Test
3445 1)src/test/java/com.example.biz.BootJpaDemoApplicationTests.java > right-click > Run As > JUnit Test
3446 2)Green bar
3447
3448
3449 6. template 준비하기
3450 1)src/main/resources > right-click > New > File
3451 2)File name : messages.properties
3452
3453 content.title=Message sample page.
3454 content.message=This is sample message from properties.
3455
3456 3)src/main/resources/static > right-click > New > Web > HTML Files
3457 4)Name : index.html
3458
3459 <!DOCTYPE HTML>
3460 <html xmlns:th="http://www.thymeleaf.org">
3461 <head>
3462 <title>top page</title>
3463 <meta charset="UTF-8" />
3464 <style>
3465 h1 { font-size:18pt; font-weight:bold; color:gray; }
3466 body { font-size:13pt; color:gray; margin:5px 25px; }
3467 pre { border: solid 3px #ddd; padding: 10px; }
3468 </style>
3469 </head>
3470 <body>
3471 <h1 th:text="#{content.title}">Hello page</h1>
3472 <pre th:text="\${data}"></pre>
3473 </body>
3474 </html>
3475
3476 7. 실행해서 접속
3477 1)저장돼있는 data가 회색 사각 틀 안에 표시된다.
3478 2)아직 아무 data가 없기 때문에 빈 배열 []라고 표시된다.
3479
3480
3481 8. Entity의 CRUD 처리하기 : form으로 data 저장하기
3482 1)index.html 수정
3483
3484 <!DOCTYPE HTML>
3485 <html xmlns:th="http://www.thymeleaf.org">
3486 <head>
3487 <title>top page</title>
3488 <meta charset="UTF-8" />
3489 <style>
3490 h1 { font-size:18pt; font-weight:bold; color:gray; }

```

3491     body { font-size:13pt; color:gray; margin:5px 25px; }
3492     tr { margin:5px; }
3493     th { padding:5px; color:white; background:darkgray; }
3494     td { padding:5px; color:black; background:#e0e0ff; }
3495     </style>
3496 </head>
3497 <body>
3498     <h1 th:text="#{content.title}">Hello page</h1>
3499     <table>
3500     <form method="post" action="/" th:object="${formModel}">
3501         <tr><td><label for="username">이름</label></td>
3502             <td><input type="text" name="username" th:value="*{username}" /></td></tr>
3503         <tr><td><label for="age">연령</label></td>
3504             <td><input type="text" name="age" th:value="*{age}" /></td></tr>
3505         <tr><td><label for="email">메일</label></td>
3506             <td><input type="text" name="email" th:value="*{email}" /></td></tr>
3507         <tr><td><label for="memo">메모</label></td>
3508             <td><textarea name="memo" th:text="*{memo}"
3509                 cols="20" rows="5"></textarea></td></tr>
3510         <tr><td></td><td><input type="submit" /></td></tr>
3511     </form>
3512 </table>
3513 <hr/>
3514 <table>
3515     <tr><th>ID</th><th>이름</th></tr>
3516     <tr th:each="obj : ${datalist}">
3517         <td th:text="${obj.id}"></td>
3518         <td th:text="${obj.username}"></td>
3519     </tr>
3520 </table>
3521 </body>
3522 </html>

```

2)HelloController.java 수정

```

3525
3526 package com.example.biz;
3527
3528 import org.springframework.beans.factory.annotation.Autowired;
3529 import org.springframework.stereotype.Controller;
3530 import org.springframework.transaction.annotation.Transactional;
3531 import org.springframework.web.bind.annotation.ModelAttribute;
3532 import org.springframework.web.bind.annotation.RequestMapping;
3533 import org.springframework.web.bind.annotation.RequestMethod;
3534 import org.springframework.web.servlet.ModelAndView;
3535
3536 import com.example.biz.dao.UserRepository;
3537 import com.example.biz.vo.User;
3538
3539 @Controller
3540 public class HelloController {
3541
3542     @Autowired

```

```
3543     UserRepository repository;
3544
3545     @RequestMapping(value = "/", method = RequestMethod.GET)
3546     public ModelAndView index(@ModelAttribute("formModel") User mydata, ModelAndView mav) {
3547         mav.setViewName("index");
3548         mav.addObject("msg","this is sample content.");
3549         Iterable<User> list = repository.findAll();
3550         mav.addObject("datalist",list);
3551         return mav;
3552     }
3553
3554     @RequestMapping(value = "/", method = RequestMethod.POST)
3555     @Transactional(readOnly=false)
3556     public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView mav) {
3557         repository.saveAndFlush(mydata);
3558         return new ModelAndView("redirect:/");
3559     }
3560 }
3561
3562 3)접속해서 실행
3563     -form에 text를 입력해서 전송하면 data가 추가돼서 아래쪽 table에 표시된다.
3564
3565
3566 9. @ModelAttribute와 data 저장
3567     1)@ModelAttribute
3568         -이것은 entity class의 instance를 자동으로 적용할 때 사용
3569         -인수에는 instance 이름을 지정한다.
3570         -이것은 전송 form에서 th:object로 지정하는 값이 된다.
3571         -전송된 form의 값이 자동으로 User instance로 저장된다.
3572         -따라서 이 annotation을 이용하면 이렇게 쉽게 전송한 data를 저장할 수 있다.
3573
3574     2)saveAndFlush() method
3575         -HomeController.java의 아래 code를 보자.
3576
3577         @RequestMapping(value = "/", method = RequestMethod.POST)
3578         @Transactional(readOnly=false)
3579         public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView mav) {
3580             repository.saveAndFlush(mydata);
3581             return new ModelAndView("redirect:/");
3582         }
3583
3584     3)미리 설정한 entity는 JpaRepository의 saveAndFlush라는 method를 통해 entity를 영구화한다.
3585     4)Database를 사용하고 있다면 Database에 그대로 저장된다.
3586
3587
3588 10. @Transactional과 transaction
3589     1)바로 위의 code에서 @Transactional(readOnly=false)가 있다.
3590     2)이 annotation은 transaction을 위한 것이다.
3591     3)이 annotation때문에 method내에서 실행되는 database 처리가 일괄적으로 실행되게 된다.
3592     4)data 변경 처리는 도중에 외부 접속에 의해 data 구조나 내용이 바뀌면 data 일관성에 문제가 발생하게 된다.
3593     5)이런 문제를 방지하기 위해 transaction이 사용되는 것이다.
```


3594 6)code를 보면 readOnly=false라고 설정하고 있다.
3595 7)이 readOnly는 문자 그대로 '읽기 전용(변경 불가)임을 의미한다.
3596 8)readOnly=false라고 설정하면 변경을 허가하는 transaction이다.
3597
3598
3599 11. Data 초기화 처리
3600 1)저장한 data는 application 오 르 종료하고 다시 실행하면 지워진다.
3601 2)HSQLDB는 기본적으로 memory내에 data를 cache하고 있으므로 종료와 함께 지워지는 것이다.
3602 3)controller에 data를 작성하는 초기화 처리를 넣기로 한다.
3603 4>HelloController.java code 추가
3604
3605 @PostConstruct
3606 public void init(){
3607 User user1 = new User();
3608 user1.setUsername("한지민");
3609 user1.setAge(24);
3610 user1.setEmail("javaexpert@nate.com");
3611 user1.setMemo("Hello, Spring JPA");
3612 repository.saveAndFlush(user1);
3613
3614 User user2 = new User();
3615 user2.setUsername("조용필");
3616 user2.setAge(66);
3617 user2.setEmail("aaa@aaa.com");
3618 user2.setMemo("Good Morning!");
3619 repository.saveAndFlush(user2);
3620
3621 User user3 = new User();
3622 user3.setUsername("이미자");
3623 user3.setAge(70);
3624 user3.setEmail("bbb@bbb.com");
3625 user3.setMemo("Spring Boot is very good.");
3626 repository.saveAndFlush(user3);
3627 }
3628
3629 5)@PostConstruct는 생성자를 통해 instance가 생성된 후에 호출되는 method임을 나타낸다.
3630 6)Controller는 처음에 한 번만 instance를 만들고 이후에는 해당 instance를 유지한다.
3631 7)따라서 여기에 test용 data 작성 처리를 해두면 application 실행시에 반드시 한 번 실행되어, data가 준비되는 것이다.
3632
3633
3634 12. User Find 및 Update 처리하기
3635 1)src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
3636 2)File name : edit.html > Finish
3637
3638 <!DOCTYPE HTML>
3639 <html xmlns:th="http://www.thymeleaf.org">
3640 <head>
3641 <title>edit page</title>
3642 <meta charset="UTF-8" />
3643 <style>
3644 h1 { font-size:18pt; font-weight:bold; color:gray; }

```

3645     body { font-size:13pt; color:gray; margin:5px 25px; }
3646     tr { margin:5px; }
3647     th { padding:5px; color:white; background:darkgray; }
3648     td { padding:5px; color:black; background:#e0e0ff; }
3649     </style>
3650 </head>
3651 <body>
3652     <h1 th:text="${title}">Edit page</h1>
3653     <table>
3654     <form method="post" action="/edit" th:object="${formModel}">
3655         <input type="hidden" name="id" th:value="{id}" />
3656         <tr><td><label for="username">이름</label></td>
3657             <td><input type="text" name="username" th:value="{username}" /></td></tr>
3658         <tr><td><label for="age">연령</label></td>
3659             <td><input type="text" name="age" th:value="{age}" /></td></tr>
3660         <tr><td><label for="email">메일</label></td>
3661             <td><input type="text" name="email" th:value="{email}" /></td></tr>
3662         <tr><td><label for="memo">메모</label></td>
3663             <td><textarea name="memo" th:text="{memo}"
3664                 cols="20" rows="5"></textarea></td></tr>
3665         <tr><td></td><td><input type="submit" /></td></tr>
3666     </form>
3667     </table>
3668 </body>
3669 </html>
3670
3671 3)UserRepository code 추가
3672
3673     @Repository("repository")
3674     public interface UserRepository extends JpaRepository<User, Long> {
3675         public User findById(Long id);
3676     }
3677
3678 4)RequestHandler 작성하기
3679     -HelloController.java code 추가
3680
3681     @RequestMapping(value = "/edit/{id}", method = RequestMethod.GET)
3682     public ModelAndView edit(@ModelAttribute User user, @PathVariable int id, ModelAndView mav)
3683     {
3684         mav.setViewName("edit");
3685         mav.addObject("title","edit mydata.");
3686         User findUser = repository.findById((long)id);
3687         mav.addObject("formModel", findUser);
3688         return mav;
3689     }
3690
3691     @RequestMapping(value = "/edit", method = RequestMethod.POST)
3692     @Transactional(readOnly=false)
3693     public ModelAndView update(@ModelAttribute User user, ModelAndView mav) {
3694         repository.saveAndFlush(user);
3695         return new ModelAndView("redirect:/");
3696     }

```

```

3696
3697 5)접속해서 아래와 같이 URL을 입력하면
3698
3699 http://localhost:8080/edit/1
3700
3701 -해당 ID의 data가 표시된다.
3702 -data를 변경하고 전송해보자.
3703 -findById는 어디서 구현되는 것일까?
3704 --repository는 method의 이름을 기준으로 entity 검색 처리를 자동 생성한다.
3705 --즉 repository에 method 선언만 작성하고, 구체적인 처리를 구현할 필요가 없다.
3706
3707
3708 13. Entity delete 구현하기
3709 1)update를 하고 select를 했으니 이번에는 delete를 해 보자.
3710 2)delete.html template를 작성한다.
3711 3)src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
3712 4)File name : delete.html > Finish
3713
3714 <!DOCTYPE HTML>
3715 <html xmlns:th="http://www.thymeleaf.org">
3716 <head>
3717 <title>edit page</title>
3718 <meta charset="UTF-8" />
3719 <style>
3720 h1 { font-size:18pt; font-weight:bold; color:gray; }
3721 body { font-size:13pt; color:gray; margin:5px 25px; }
3722 td { padding:0px 20px; background:#eee;}
3723 </style>
3724 </head>
3725 <body>
3726 <h1 th:text="${title}">Delete page</h1>
3727 <table>
3728 <form method="post" action="/delete" th:object="${formModel}">
3729 <input type="hidden" name="id" th:value="*{id}" />
3730 <tr><td><p th:text="|이름 : *{username}|"></p></td></tr>
3731 <tr><td><p th:text="|연령 : *{age}|" ></p></td></tr>
3732 <tr><td><p th:text="*{email}"></p></td></tr>
3733 <tr><td><p th:text="*{memo}"></p></td></tr>
3734 <tr><td><input type="submit" value="delete"/></td></tr>
3735 </form>
3736 </table>
3737 </body>
3738 </html>
3739
3740 5)RequestHandler 작성
3741
3742 @RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
3743 public ModelAndView delete(@PathVariable int id, ModelAndView mav) {
3744     mav.setViewName("delete");
3745     mav.addObject("title","delete mydata.");
3746     User user = repository.findById((long)id);
3747     mav.addObject("formModel",user);

```

```
3748         return mav;
3749     }
3750
3751     @RequestMapping(value = "/delete", method = RequestMethod.POST)
3752     @Transactional(readOnly=false)
3753     public ModelAndView remove(@RequestParam long id, ModelAndView mav) {
3754         repository.delete(id);
3755         return new ModelAndView("redirect:/");
3756     }
3757
3758 6)접속해서 실행
3759 http://localhost:8080/delete/2라고 하면 id가 2번이 출력되고 delete button을 누르면 삭제된다.
```