```
1 HOL: Spring Boot
 2
 3 Task1. Maven으로 Spring Boot Project 생성하기
 4 1. In Package Explorer
     1)right-click > New > Project > Maven > Maven Project > Next
 5
     2)[New Maven project] 창에서 > Next
 6
 8
     3)Select an Archetype
 9
       -Group Id: org.apache.maven.archetypes
10
       -Artifact Id: maven-archetype-quickstart
11
       -Version: 1.4
12
       -Next
13
14
     4)Enter an artifact id
15
       -Group Id: com.example
       -Artifact Id: springbootdemo
16
17
       -Version: 0.0.1-SNAPSHOT
18
       -Package: com.example.springbootdemo
19
       -Finish
20
21
22
   2. pom.xml 수정
     <?xml version="1.0" encoding="UTF-8"?>
23
24
25
     project xmlns="http://maven.apache.org/POM/4.0.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
26
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
27
       http://maven.apache.org/xsd/maven-4.0.0.xsd">
28
       <modelVersion>4.0.0</modelVersion>
29
30
       <groupId>com.example</groupId>
31
       <artifactId>springbootdemo</artifactId>
       <version>0.0.1-SNAPSHOT</version>
32
33
34
       <name>springbootdemo</name>
35
       <!-- FIXME change it to the project's website -->
       <url><ahref="http://www.example.com">url><ahref="http://www.example.com">http://www.example.com</a></url>
36
37
       <parent>
38
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-parent</artifactId>
39
         <version>2.2.6.RELEASE
40
41
       </parent>
42
43
       properties>
44
         45
         <maven.compiler.source>13</maven.compiler.source>
46
         <maven.compiler.target>13</maven.compiler.target>
47
       </properties>
48
49
       <dependencies>
50
         <dependency>
51
           <groupId>junit</groupId>
52
           <artifactId>junit</artifactId>
53
           <version>4.13</version>
54
           <scope>test</scope>
55
         </dependency>
         <dependency>
56
57
           <groupId>org.springframework.boot</groupId>
```

```
58
             <artifactId>spring-boot-starter-web</artifactId>
 59
           </dependency>
 60
           <dependency>
 61
             <groupId>org.springframework.boot</groupId>
 62
            <artifactId>spring-boot-starter-test</artifactId>
 63
             <scope>test</scope>
 64
           </dependency>
         </dependencies>
 65
 66
 67
        . . .
 68
 69
       1)<parent> 설정하기
 70
        -Spring Boot의 설정 정보를 상속한다.
 71
        -여기서 지정한 version이 spring boot의 version이 된다.
        -spring boot의 version을 올리려면 <version> tag 안에 있는 설정 값을 변경한다.
 72
 73
 74
       2)spring-boot-starter-web
 75
        -spring boot로 web application을 만들 때 참조할 기본 library 정보를 설정한다.
 76
        -이렇게 쓰기만 해도 web application 제작에 필요한 spring framework 관련 library와 third-party
        library를 이용할 수 있게 된다.
 77
        -version은 위 parent에서 설정한 spring-boot-starter-parent 안에 정의되어 있으므로, 여기서는 지정하지
        않아도 된다.
 78
 79
       3)pop.xml > right-click > Run As > Maven install
 80
 81
 82
    3. Project > right-click > Properties
 83
       1)Java Build Path > Modulepath > JRE System Library [jdk-13.0.2] > Apply
       2) Java Compiler > JDK Compliance > 13 > Apply
 84
 85
       3)Project Facets > Java 13 > Runtimes tab > Check [jdk-13.0.2]
       4)Apply and Close
 86
 87
 88
 89 4. Project > right-click > Maven > Update Project... > OK
 90
 91
 92
    5. Hello World!를 출력하는 Web application 작성하기
 93
       1)src/main/java/com/example/springbootdemo/App.java
 94
 95
        package com.example.springbootdemo;
 96
 97
        import org.springframework.boot.SpringApplication;
 98
        import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
 99
        import org.springframework.web.bind.annotation.RequestMapping;
        import org.springframework.web.bind.annotation.RestController;
100
101
102
        * Hello world!
103
104
        */
105
        @RestController
106
107
        @EnableAutoConfiguration
        public class App {
108
           @RequestMapping("/")
109
110
          String home(){
            return "Hello, World!";
111
112
           }
113
```

```
114
          public static void main( String[] args ){
115
           SpringApplication.run(App.class, args);
116
         }
117
        }
118
119
      2)@RestController
120
        -이 annotation을 붙이면 web appication에서 request을 받아들이는 controller class임을 나타낸다.
121
122
      3)@EnableAutoConfiguration
123
        -이 annotation은 매우 중요하다.
124
        -이 annotation을 붙이면 다양한 설정이 자동으로 수행되고 기존의 spring application에 필요했던 설정 file들
        이 필요없게 된다.
125
126
      4)@RequestMapping("/")
        -이 annotation이 붙으면 이 method가 HTTP 요청을 받아들이는 method임을 나타낸다.
127
128
        -@GetMapping도 가능
129
        -@RequestMapping(value="/", method=RequestMethod.GET)과 @GetMapping은 동일하다.
130
131
      5)return "Hello World!";
132
        -HTTP 응답을 반환한다.
133
        -@RestController annotation이 붙은 class에 속한 method에서 문자열을 반환하면 해당 문자열이 그대로
        HTTP 응답이 되어 출력된다.
134
135
      6)SpringApplication.run(App.clas, args);
        -spring boot application을 실행하는 데 필요한 처리를 main() 안에서 작성한다.
136
137
        -@EnableAutoConfiguration annotation이 붙은 class를 SpringApplication.run()의 첫번째 인자로
        지정한다.
138
139
140 6. Web Application 실행하기
      1)springbootdemo project > right-click > Run As > Spring Boot App
141
142
143
144
        145
         146
147
          148
        ======|_|=======|___/=/_/_/
149
         :: Spring Boot ::
                         (v2.2.6.RELEASE)
150
        2020-04-20 21:58:03.208 INFO 14596 --- [
151
                                                     main]
                                            : Starting App on DESKTOP-1BKHISM with PID
        com.example.springbootdemo.App
        14596 (C:\SpringHome\springbootdemo\target\classes started by devex in
        C:\SpringHome\springbootdemo)
152
        2020-04-20 21:58:03.210 INFO 14596 --- [
                                                     main]
        com.example.springbootdemo.App
                                            : No active profile set, falling back to default
        profiles: default
153
        2020-04-20 21:58:03.763 INFO 14596 --- [
                                                     main]
        o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat initialized with port(s): 8080
154
        2020-04-20 21:58:03.771 INFO 14596 --- [
                                                     main]
        o.apache.catalina.core.StandardService : Starting service [Tomcat]
        2020-04-20 21:58:03.771 INFO 14596 --- [
155
                                                     main]
        org.apache.catalina.core.StandardEngine: Starting Servlet engine: [Apache
        Tomcat/9.0.33]
156
        2020-04-20 21:58:03.837 INFO 14596 --- [
        o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded
        WebApplicationContext
```

```
157
        2020-04-20 21:58:03.838 INFO 14596 --- [
                                                          main] o.s.web.context.ContextLoader
                : Root WebApplicationContext: initialization completed in 602 ms
        2020-04-20 21:58:03.980 INFO 14596 --- [
158
                                                          main]
        o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService
        'applicationTaskExecutor'
159
        2020-04-20 21:58:05.842 INFO 14596 --- [
                                                          main1
        o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on port(s): 8080 (http)
        with context path "
160
        2020-04-20 21:58:05.845 INFO 14596 --- [
                                                          main]
        com.example.springbootdemo.App : Started App in 2.846 seconds (JVM running
        for 5.125)
161
162
       2)출력된 log 내용을 보면 8080 port로 tomcat이 시작된다는 것을 알 수 있다.
       3)SpringApplication.run() method에서 내장 server를 시작했기 때문이다.
163
       4)http://localhost:8080/로 접속해보자.
164
       5)Web browser에 'Hello, World!'가 출력된다.
165
166
       6)Console에는 아래와 같이 출력된다.
        2020-04-20 21:59:13.181 INFO 14596 --- [nio-8080-exec-1]
167
        o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet
        'dispatcherServlet'
168
        2020-04-20 21:59:13.181 INFO 14596 --- [nio-8080-exec-1]
        o.s.web.servlet.DispatcherServlet
                                           : Initializing Servlet 'dispatcherServlet'
        2020-04-20 21:59:13.186 INFO 14596 --- [nio-8080-exec-1]
169
        o.s.web.servlet.DispatcherServlet
                                            : Completed initialization in 5 ms
170
       7)application을 끝내려면 Ctrl + C를 누르고, '[일괄 작업을 끝내시겠습니까 (Y/N)?'라는 질문에 'y'를 입력하고
171
       enter key를 누르면 된다.
172
        -또는 빨간색 실행 중지 Button을 click한다.
           2020-04-20 21:59:47.560 INFO 14596 --- [on(8)-127.0.0.1]
173
          inMXBeanRegistrar$SpringApplicationAdmin: Application shutdown requested.
           2020-04-20 21:59:47.562 INFO 14596 --- [on(8)-127.0.0.1]
174
          o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService
           'applicationTaskExecutor'
175
176
       8)여기서 알게 된 사실
        -설정할 의존 관계의 갯수가 적다.
177
        -Java Class 하나만 작성하면 된다.
178
        -명령 prompt에서 application을 실행한다.
179
180
181
182
183 ---
184 Task2. STS로 Spring Boot Application 개발하기
    1. Package Explorer > right-click > New > Spring Starter Project
185
       1)Service URL: <a href="http://start.spring.io">http://start.spring.io</a>
186
187
       2)Name: demo
188
       3)Type: Maven
       4)Packaging: jar
189
       5) Java Version: 8
190
191
       6)Language: Java
192
       7) Group: com.example
       8)Artifact: demo
193
194
       9) Version: 0.0.1-SNAPSHOT
195
       10)Description: Demo project for Spring Boot
196
       11)Package: com.example.demo
197
       12)Next
198
199
```

```
200 2. [New Spring Starter Project Dependencies] 창에서
      1) Spring Boot Version: 2.2.6
201
202
      2)Select Web > check Spring Web > Finish
203
204
205 3. src/main/java/com.example.demo.DemoApplication.java
206
207
      package com.example.demo;
208
209
      import org.springframework.boot.SpringApplication;
210
      import org.springframework.boot.autoconfigure.SpringBootApplication;
211
212
      @SpringBootApplication
213
      public class DemoApplication {
214
215
        public static void main(String[] args) {
216
          SpringApplication.run(DemoApplication.class, args);
217
218
      }
219
220
221 4. DemoApplication.java > right-click > Run As > Spring Boot App
222
223
224
      225
226
          ___)| |_)| | | | | (_| | ) ) )
227
228
             __| .__|_| |_|_| |_\__, | / / / /
       ======|_|=======|___/=/_/_/
229
230
       :: Spring Boot ::
                           (v2.2.6.RELEASE)
231
232
      2020-04-20 22:03:17.462 INFO 15496 --- [
                                                      main]
      com.example.demo.DemoApplication : Starting DemoApplication on
      DESKTOP-1BKHISM with PID 15496 (C:\SpringHome\demo\target\classes started by devex
      in C:\SpringHome\demo)
      2020-04-20 22:03:17.464 INFO 15496 --- [
233
                                                      main]
      com.example.demo.DemoApplication : No active profile set, falling back to default
      profiles: default
      2020-04-20 22:03:18.048 INFO 15496 --- [
234
                                                      main1
      o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat initialized with port(s): 8080 (http)
235
      2020-04-20 22:03:18.055 INFO 15496 --- [
                                                      main]
      o.apache.catalina.core.StandardService : Starting service [Tomcat]
236
      2020-04-20 22:03:18.055 INFO 15496 --- [
                                                      main]
      org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
      Tomcat/9.0.331
237
      2020-04-20 22:03:18.120 INFO 15496 --- [
                                                      main] o.a.c.c.C.[Tomcat].[localhost].[/]
          : Initializing Spring embedded WebApplicationContext
      2020-04-20 22:03:18.120 INFO 15496 --- [
238
                                                      main] o.s.web.context.ContextLoader
            : Root WebApplicationContext: initialization completed in 626 ms
239
      2020-04-20 22:03:18.246 INFO 15496 --- [
                                                      main]
      o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService
      'applicationTaskExecutor'
240
      2020-04-20 22:03:20.066 INFO 15496 --- [
                                                      main]
      o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on port(s): 8080 (http) with
      context path "
      2020-04-20 22:03:20.069 INFO 15496 --- [
241
                                                      main]
      com.example.demo.DemoApplication : Started DemoApplication in 2.811 seconds
```

```
(JVM running for 5.007)
242
243
244 5. http://localhost:8080
245
      Whitelabel Error Page
246
      This application has no explicit mapping for /error, so you are seeing this as a fallback.
247
248
      Mon Apr 20 22:04:04 KST 2020
249
      There was an unexpected error (type=Not Found, status=404).
250
      No message available
251
252
253 6. src/main/java/com.example.demo.DemoApplication.java 수정하기
254
255
      package com.example.demo;
256
257
      import org.springframework.boot.SpringApplication;
258
      import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
      import org.springframework.web.bind.annotation.RequestMapping;
259
260
      import org.springframework.web.bind.annotation.RestController;
261
262
      @RestController
263
      @EnableAutoConfiguration
264
      public class DemoApplication {
265
266
        @RequestMapping("/")
        String home() {
267
268
          return "Hello, World!";
269
270
271
        public static void main(String[] args) {
272
          SpringApplication.run(DemoApplication.class, args);
273
274
275
      }
276
277
      1)빨간색 실행 중비 button click
278
      2)DemoApplication.java > right-click > Run As > Spring Boot App
279
      3)http://localhost:8080/
280
        Hello, World!
281
282
283
284 -----
285 Task3. Groovy로 Application 개발하기
286 1. 준비
287
      1)Visit
        https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-
288
        spring-boot.html
289
290
      2)3.2.1 Manual Installation에서 spring-boot-cli-2.2.4.RELEASE-bin.zip link를 click한다.
291
      3)Unzip > Move to C:\Program Files\spring-2.2.4.RELEASE
292
      4)path 설정
293
        -%PATH%;C:\Program Files\spring-2.2.1.RELEASE\bin
294
295
296 2. Groovy Script 작성하기
297
      1)Editor(예:VSCode)를 열어서 아래의 code를 적당한 위치(즉 C:\temp)에 file 이름은 app.groovy라고 저
```

```
장한다.
298
      2)app.groovy
299
300
        @RestController
301
        class App {
302
303
          @RequestMapping("/")
          def home() {
304
305
            "Hello!!!"
306
307
        }
308
309
      3)app.groovy 실행하기
310
        -Command Prompt에서,
311
        $ cd C:\temp
312
        $ spring run app.groovy
313
        Resolving dependencies.....
314
315
316
         /\\ / ___
        317
         318
319
               _| ._|_| |_| |_\__, | / / / /
320
         ======|_|=======|__/=/_/_/
                            (v2.2.4.RELEASE)
321
         :: Spring Boot ::
322
323
        2020-02-18 23:22:18.024 INFO 12048 --- [
                                                     runner-0] o.s.boot.SpringApplication
              : Starting application on DESKTOP-1BKHISM with PID 12048 (started by devex in
        C:\Temp)
324
        2020-02-18 23:22:18.039 INFO 12048 --- [
                                                     runner-0] o.s.boot.SpringApplication
              : No active profile set, falling back to default profiles: default
325
        2020-02-18 23:22:22.487 INFO 12048 --- [
                                                     runner-01
        o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat initialized with port(s): 8080
        (http)
326
        2020-02-18 23:22:22.531 INFO 12048 --- [
                                                     runner-01
        o.apache.catalina.core.StandardService : Starting service [Tomcat]
        2020-02-18 23:22:22.532 INFO 12048 --- [
327
                                                     runner-0]
        org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
        Tomcat/9.0.301
        2020-02-18 23:22:22.539 INFO 12048 --- [
328
                                                     runner-01
        o.a.catalina.core.AprLifecycleListener : Loaded APR based Apache Tomcat Native library
        [1.2.23] using APR version [1.7.0].
329
        2020-02-18 23:22:22.540 INFO 12048 --- [
                                                     runner-01
        o.a.catalina.core.AprLifecycleListener : APR capabilities: IPv6 [true], sendfile [true],
        accept filters [false], random [true].
330
        2020-02-18 23:22:22.541 INFO 12048 --- [
                                                     runner-0]
        o.a.catalina.core.AprLifecycleListener : APR/OpenSSL configuration: useAprConnector
        [false], useOpenSSL [true]
        2020-02-18 23:22:22.552 INFO 12048 --- [
331
                                                     runner-01
        o.a.catalina.core.AprLifecycleListener : OpenSSL successfully initialized [OpenSSL 1.1.1c
         28 May 2019]
        2020-02-18 23:22:22.667 INFO 12048 --- [
332
                                                     runner-0]
        org.apache.catalina.loader.WebappLoader : Unknown class loader
        [org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentC
        lassLoader@6e84d437] of class [class
        org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentCl
        assLoader]
333
        2020-02-18 23:22:22.800 INFO 12048 --- [
                                                     runner-0]
```

```
o.a.c.c.C.[Tomcat].[localhost].[/]
                                           : Initializing Spring embedded
        WebApplicationContext
334
        2020-02-18 23:22:22.801 INFO 12048 --- [
                                                      runner-0]
        o.s.web.context.ContextLoader
                                            : Root WebApplicationContext: initialization
        completed in 4158 ms
335
        2020-02-18 23:22:23.454 INFO 12048 --- [
                                                      runner-01
        o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService
        'applicationTaskExecutor'
336
        2020-02-18 23:22:24.919 INFO 12048 --- [
                                                      runner-0]
        o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on port(s): 8080 (http)
        with context path "
        2020-02-18 23:22:24.929 INFO 12048 --- [
                                                      runner-0] o.s.boot.SpringApplication
337
              : Started application in 9.024 seconds (JVM running for 17.243)
338
339
340 3. Web browser로 http://localhost:8080에 접속한다.
341
      Hello!!!
342
343
344 4. app.groovy code 수정하기
345
      1)Command Prompt 에서 Ctrl + C를 눌러서 종료시킨다.
346
      2)일괄 작업을 끝내시겠습니까 (Y/N)? Y
347
      3) 아래와 같이 code를 수정한다.
348
349
        @RestController
350
        class App {
351
352
          @RequestMapping("/")
          def home() {
353
            def header = "<html><body>"
354
            def footer = "</body></html>"
355
356
            def content = "<h1>Hello! Spring Boot with Groovy</h1>This is html
            content."
357
358
            header + content + footer
359
          }
        }
360
361
362
363
    5. 다시 script를 실행한다.
364
      $ spring run app.groovy
365
366
367 6. Browser를 refresh 한다.
368
      Hello! Spring Boot with Groovy
369
370
      This is html content.
371
372
373
    7. Template 사용하기
374
      1)template은 HTML을 기반으로 작성된 code를 읽어 rendering해서 web page에 출력하는 기능이다
375
      2)이런 기능의 template이 몇 가지 종류가 있지만, spring boot에서는 thymeleaf(타임리프)라고 하는 library
      를 자주 사용한다.
376
      3)http://www.thymeleaf.org
377
      4)template file 작성
378
      5)C:\temp\templates\home.html
379
380
        <!doctype html>
```

```
381
         <html lang="en">
382
383
         <head>
384
           <meta charset="UTF-8"/>
385
           <title>Index Page</title>
386
           <style type="text/css">
387
              h1 {
388
                font-size: 18pt;
389
                font-weight: bold;
390
                color: gray;
391
              }
392
393
              body {
394
                font-size: 13pt;
395
                color: gray;
396
                margin: 5px 25px;
397
398
           </style>
399
         </head>
400
401
         <body>
402
           <h1>Hello! Spring Boot with Thymeleaf</h1>
403
           This is sample web page.
404
         </body>
405
         </html>
406
407
408
       6)template file은 controller가 있는 곳의 templates folder 안에 두어야 한다.
409
       7)controller 수정하기
410
411
         -app.groovy
412
           @Grab("thymeleaf-spring5")
413
414
           @Controller
415
          class App {
416
            @RequestMapping("/")
417
418
            @ResponseBody
419
            def home(ModelAndView mav) {
420
              mav.setViewName("home")
421
              mav
422
            }
423
           }
424
425
       8)다시 script 실행
426
         $ spring run app.groovy
427
428
       9)http://localhost:8080
429
        Hello! Spring Boot with Thymeleaf
430
431
         This is sample web page.
432
433
434 8. form 전송하기
435
       1)home.html
436
437
         <!doctype html>
438
         <html lang="en">
```

```
439
           <head>
             <meta charset="UTF-8" />
440
             <title>Index Page</title>
441
442
             <style type="text/css">
443
               h1 { font-size:18pt; font-weight:bold; color:gray; }
444
               body { font-size:13pt; color:gray; margin:5px 25px; }
445
             </style>
           </head>
446
447
           <body>
448
             <h1>Hello!</h1>
             ${msg}
449
             <form method="post" action="/send">
450
451
               <input type="text" name="text1" th:value="${value}" />
               <input type="submit" value="Send" />
452
453
454
           </body>
455
         </html>
456
457
       2)app.groovy
458
         @Grab("thymeleaf-spring5")
459
         @Controller
460
         class App {
461
462
           @RequestMapping(value = "/", method=RequestMethod.GET)
463
           @ResponseBody
464
           def home(ModelAndView mav) {
465
             mav.setViewName("home")
466
             mav.addObject("msg", "Please write your name...")
467
             mav
468
           }
           @RequestMapping(value = "/send", method=RequestMethod.POST)
469
470
           @ResponseBody
471
           def send(@RequestParam("text1") String str, ModelAndView mav){
472
              mav.setViewName("home")
473
              mav.addObject("msg", "Hello, " + str + "!!!")
474
              mav.addObject("value", str)
475
              mav
476
477
         }
478
479
       3)script 실행
480
         $ spring run app.groovy
481
482
       4)http://localhost:8080
483
         Hello!
484
         Please write your name...
485
486
         Hello, 한지민!!!
487
488
489 -----
490 Task4. SPRING INITIALIZR(Maven)
491 1. Visit <a href="http://start.spring.io/">http://start.spring.io/</a>
492 2. 설정
493
       1)Maven Project
494
       2)Java
495
       3)2.2.6
496
      4)Group: com.example
```

```
497
       5)Artifact: demo
498
       6)Name: demo
499
       7) Description: Demo project for Spring Boot
500
       8)Package Name: com.example.demo
501
       9)Packaging: Jar
502
       10)Java Version: 8
503
       11) Dependencies: [ADD DEPENDENCIES] click > Spring Web
504
505
       12)Click [Generate]
506
       13)Downloads [demo.zip]: 55.5KB
507
       14) Unpack to Spring workspace.
508
509
510 3. Project Import
511
       1)In Package Explorer > right-click > Import > Maven > Existing Maven Projects > Next
       2)Click [Browse...] > demo Folder Select > Finish
512
513
514
515 4. JUnit Test
516
       1)src/test/java/com.example.demo.DemoApplicationTests.java > right-click > Run As >
       JUnit Test >
517
         Green bar
518
519
520 5. Spring Boot App 실행하기
521
       1)demp project > right-click > Run As > Spring Boot App
522
523
       2)http://localhost:8080/
524
525
         Whitelabel Error Page
526
527
         This application has no explicit mapping for /error, so you are seeing this as a fallback.
528
         Thu Nov 07 15:47:32 KST 2019
529
         There was an unexpected error (type=Not Found, status=404).
530
         No message available
531
532
533 6. Controller 생성
534
       1)src/main/java/com.example.demo > right-click > New > Class
535
       2)Name: HelloController
536
       3)Finish
537
538
         package com.example.demo;
539
540
         import org.springframework.web.bind.annotation.GetMapping;
541
         import org.springframework.web.bind.annotation.RestController;
542
543
         @RestController
544
         public class HelloController {
545
546
           @GetMapping("/")
547
           public String hello() {
548
             return "Hello, Spring Boot World";
549
           }
550
         }
551
552
553 7. Relaunch demo
```

```
554
      -http://localhost:8080/
555
556
        Hello, Spring Boot World
557
558
559
    8. RestController 사용하기
560
      1)HelloController.java 수정하기
561
562
        @RestController
563
        public class HelloController {
564
565
          @GetMapping("/hello")
566
          public String hello(String name) {
            return "Hello!: " + name;
567
568
          }
        }
569
570
571
      2)@RestController
572
        -이 annotation을 붙이면 web appication에서 request을 받아들이는 controller class임을 나타낸다.
573
        -Spring 4부터 지원
574
        -REST 방식의 응답을 처리하는 Controller를 구현할 수 있다.
575
        -@Controller를 사용할 때 method의 return type이 문자열일 경우, 문자열에 해당하는 View를 만들어야 하
        지만, Controller를 RestController를 사용할 경우에는 Return되는 문자열이 Browser에 그대로 출력되기 때
        문에 별도로 View 화면을 만들 필요가 없다.
576
577
      3)Relaunch demo
        -http://localhost:8080/hello?name=한지민
578
579
        Hello: 한지민
580
581
582 9. VO 사용하기
583
      1)pom.xml 수정
584
        -lombok library 추가하기
585
        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
586
        <dependency>
587
           <groupId>org.projectlombok</groupId>
588
           <artifactId>lombok</artifactId>
589
           <version>1.18.12</version>
590
           <scope>provided</scope>
591
        </dependency>
592
593
        -pom.xml > right-click > Run As > Maven install
594
595
      2)com.example.demo/DemoApplication.java 수정하기
596
597
        @SpringBootApplication
        @ComponentScan(basePackages = {"com.example"})
598
599
        public class DemoApplication {
600
        }
601
602
603
      2)com.example.vo package 생성
604
      3)com.example.vo.UserVO Class 생성
605
606
        package com.example.vo;
607
608
        import lombok.Data;
609
        import lombok.AllArgsConstructor;
```

```
610
        import lombok.NoArgsConstructor;
611
612
        @Data
613
        @AllArqsConstructor
614
        @NoArgsConstructor
615
        public class UserVO {
616
          private String userid;
617
          private String name;
618
          private String gender;
619
          private String city;
620
621
622
623 10. UserController 생성하기
624
      1)com.example.controller package 생성
625
      2)com.example.controller.UserController Class 생성
626
627
        package com.example.controller;
628
629
        import org.springframework.web.bind.annotation.GetMapping;
630
        import org.springframework.web.bind.annotation.RestController;
631
632
        import com.example.vo.UserVO;
633
634
        @RestController
635
        public class UserController {
          @GetMapping("/getUser")
636
637
          public UserVO getUser() {
638
            UserVO user = new UserVO();
639
            user.setUserid("jimin");
            user.setName("한지민");
640
641
            user.setGender("여");
642
            user.setCity("서울");
643
            return user;
644
          }
645
        }
646
647
      3)Relaunch demo
648
        -http://localhost:8080/getUser
        {"userid":"jimin","name":"한지민","gender":"여","city":"서울"}
649
650
651
652
    11. Spring DevTools 사용하기
      1)위처럼 Controller에 새로운 Method가 추가되면 반드시 실행 중인 Application을 중지하고 Application을 재
653
      실행해야 한다.
654
      2)그렇게 해야만 수정된 Controller가 반영되기 때문이다.
655
      3)그렇게 반복적인 작업을 하지 않고, 즉 Controller가 수정할 때마다 매번 Application을 재실행하는 것이 번거로
      우면 Spring DevTools 기능을 이용하면 된다.
      4)현재 사용중인 Project에 DevTools를 추가하려면 pom.xml에 추가 Dependency를 추가해야 한다.
656
      5)pom.xml을 열어서 <dependency> 제일 마지막 Tag 밑에 Ctrl + Space 를 누른다.
657
      6)Context Menu에서 [Edit Starters...]를 double-click한다.
658
659
      7)[Pom file needs savings] 창에서 pom.xml 파일의 수정 내용을 저장할 것인지 물어보면 [Save Pom]
      button을 클릭한다.
660
      8)[Edit Spring Boot Starters]창에서 Developer Tools > Spring Boot DevTools 체크한다.
661
      9)그리고 [OK] 클릭한다.
662
      10)그러면 pom.xml에 다음과 같은 Code가 추가된다
663
664
        <dependency>
```

```
665
          <groupId>org.springframework.boot</groupId>
666
          <artifactId>spring-boot-devtools</artifactId>
667
          <scope>runtime</scope>
668
        </dependency>
669
670
      11)방금 추가된 DevTools 를 적용하기 위해 Application을 다시 실행한다.
671
      12)Controller의 Code를 수정하면 자동으로 Restart가 일어난다.
672
      13)Browser에서 Refresh를 누르면 수정된 Code가 반영된다.
673
      14)즉, Java Code를 수정한 뒤 Application을 다시 수동으로 시작하지 않아도 된다
674
675
676 12. Lombok Library 사용하기
677
      1)보통 VO Class를 사용할 때 Table의 Column 이름과 같은 이름을 사용한다.
678
      2)getter / setter method도 생성하고 toString() method도 생성한다.
      3)하지만 code가 지저분해지고 모든 VO class와 JPA에서 사용할 Domain Class에 이런 Method를 반복적으로
679
      작성하는 일은 사실 번거로운 일이다.
680
      4)이런 문제를 간단하게 해결하기 위한 Library가 Lombok이다.
681
      5)Lombok을 사용하면 Java File을 Compile할 때, 자동으로 생성자, getter / setter, toString() 같은
      code들을 추가해준다.
682
      6)현재 사용하고 있는 project에 Lombok Library를 추가해 보자.
683
      7)위처럼 pom.xml의 <dependency> tag 제일 마지막에 Ctrl + Space 단축키를 누른다.
684
      8)Context Menu에서 [Edit Starters...]를 double-click한다.
685
      9)[Pom file needs savings] 창에서 pom.xml 파일의 수정 내용을 저장할 것인지 물어보면 [Save Pom]
      button을 클릭한다.
      10)[Edit Spring Boot Starters]창에서 Developer Tools > Lombok 체크한다.
686
      11) 그리고 [OK] 클릭한다.
687
688
      12)그러면 pom.xml에 다음과 같은 Code가 추가된다.
689
690
        <dependency>
691
          <groupId>org.projectlombok</groupId>
692
          <artifactId>lombok</artifactId>
693
        </dependency>
694
695
      13)Lombok을 사용하려면 별도로 STS 설치 Folder에 Lombok Library를 추가해야 한다.
696
      14)STS에 Lombok Library를 추가하기 위해 STS를 일단 종료한다.
697
      15)Lombok Homepage(https://projectlombok.org/)를 방문한다.
698
      16)download page로 이동하여 현재 최신 버전인 1.18.10을 Downloads 한다.
699
      17)download 한 Folder로 이동하여 Cmd 창에서 아래의 명령을 수행한다.
700
701
        java -jar lombok.jar
702
703
      18)[Project Lombok v1.18.10 - Installer] 창에서, IDEs에 보면 현재 Eclipse와 STS가 설치된 folder가
      자동감지된다.
704
      19)확인이 되었으면 [Install/Update] button click한다.
705
      20)[Quit Installer] button click 한다.
706
      21)Lombok이 설치되면 STS 설치 Folder(C:\Program Files\sts-4.4.0.RELEASE)에 lombok.jar가 있
      는 것을 확인할 수 있다.
707
      22)다시 STS를 실행하여 UserVO.iava로 들어간다.
708
709
        package com.example.vo;
710
711
        import lombok. Getter;
712
        import lombok.Setter;
713
        import lombok.ToString;
714
715
        @Getter
716
        @Setter
717
        @ToString
```

```
718
        public class UserVO {
           private String userid;
719
720
           private String name;
721
          private String gender;
722
          private String city;
723
        }
724
725
       23)수정된 UserVO.java 를 저장하고 왼쪽의 Package Explorer에서 UserVO.java의 하위를 클릭하면
       Getter / Setter, toString() 이 자동으로 추가된 것을 확인할 수 있다.
726
       24)다음은 Lombok에서 제공하는 Annotation이다.
727
        -@Getter
728
          -- Getter Method 생성
729
        -@Setter
730
          --Setter Method 생성
731
        -@RequiredArasConstructor
          --모든 Member 변수를 초기화하는 생성자를 생성
732
733
        -@ToString
734
           --모든 Member 변수의 값을 문자열로 연결하여 리턴하는 toString() 메소드 생성
735
        -@EqualsAndHashCode
736
          --equals(), hashCode() Method 생성
737
        -@Data
738
           --@Getter, @Setter, @RequiredArgsConstructor, @ToString, @EquqlsAndHashCode 모두
           생성.
739
740
741
742
743 Task5. SPRING INITIALIZR(Gradle)
744 1. Visit <a href="http://start.spring.io/">http://start.spring.io/</a>
745 2. 설정
       1)Gradle Project
746
747
       2)Java
748
       3)2.2.4
749
      4) Group: com.example
750
       5)Artifact: demoweb
       6)Name: demoweb
751
       7) Description: Demo project for Spring Boot
752
       8)Package Name: com.example.demoweb
753
754
       9)Packaging: Jar
755
       10)Java Version: 8
756
       11)dependencies: Developer Tools > SpringBoot DevTools, Web > Web
757
758
       12)Click [Generate]
759
       13)Downloads [demoweb.zip]: 57.8KB
760
       14) Unpack to Spring workspace.
761
762
763 3. Project Import
       1)In STS, Package Explorer > right-click > Import > Gradle > Existing Gradle Project >
764
       Next > Next
       2)Click [Browse...] > demoweb Folder > Select Folder > Next > Finish
765
766
      3)build.gradle
767
768
        plugins {
769
          id 'org.springframework.boot' version '2.2.4.RELEASE'
770
          id 'io.spring.dependency-management' version '1.0.9.RELEASE'
771
          id 'java'
772
        }
```

```
774
         group = 'com.example'
         version = '0.0.1-SNAPSHOT'
775
776
         sourceCompatibility = '1.8'
777
778
         configurations {
779
           developmentOnly
780
           runtimeClasspath {
781
             extendsFrom developmentOnly
782
           }
783
         }
784
785
         repositories {
786
           mavenCentral()
787
788
789
         dependencies {
790
           implementation 'org.springframework.boot:spring-boot-starter-web'
791
           developmentOnly 'org.springframework.boot:spring-boot-devtools'
792
           testImplementation('org.springframework.boot:spring-boot-starter-test') {
793
             exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
794
795
         }
796
797
         test {
798
           useJUnitPlatform()
799
         }
800
801
       4)src/test/java/com.example.demoweb.DemowebApplicationTests.java > right-click > Run
       As > JUnit Test > Green bar
802
803
       5)demoweb Project > right-click > Run As > Spring Boot App
804
       6)http://localhost:8080/
805
806
         Whitelabel Error Page
807
808
         This application has no explicit mapping for /error, so you are seeing this as a fallback.
         Thu Nov 07 16:06:13 KST 2019
809
810
         There was an unexpected error (type=Not Found, status=404).
811
         No message available
812
813
814
    4. Controller 생성
815
       1)src/main/java/com.example.demoweb > right-click > New > Class
816
       2)Name: HomeController
817
818
         package com.example.demoweb;
819
         import org.springframework.web.bind.annotation.GetMapping;
820
821
         import org.springframework.web.bind.annotation.RestController;
822
823
         @RestController
824
         public class HomeController {
825
826
           @GetMapping("/")
827
           public String home() {
828
             return "Hello, Spring Boot World";
829
           }
```

```
830
        }
831
832
      3)Relaunch demo
833
      4)http://localhost:8080/
        Hello, Spring Boot World
834
835
836
837
838
839 Task6. 사용자 정의 Starter 만들기
840 1. Maven Project 생성
841
      1)Project Explorer > right-click > New > Maven Project
842
      2)Next
843
      3)org.apache.maven.archetypes, maven-archetype-quickstart, 1.4 > Next
844
      4) Group Id: com.example
        -Artifact Id: mybootstarter
845
846
        -Version: 0.0.1-SNAPSHOT
847
        -Package: com.example.mybootstarter
848
        -Finish
849
850
851 2. Project Facets 수정하기
      1)mybootstarter Project > right-click > Properties > Project Facets
852
853
      2)Java 1.8 > Runtimes Tab > Apache Tomcat v9.0, jdk1.8.0 241 check
      3)Apply and Close click
854
855
856
857
    3. Mvnrepository에서 'spring boot'로 검색
858
      1)Spring Boot Autoconfigure > 2.2.4.RELEASE
859
      2)아래 코드 복사 후 pom.xml 에 붙여넣기
860
861
        https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-autoconfigure
        -->
862
        <dependency>
          <groupId>org.springframework.boot</groupId>
863
          <artifactId>spring-boot-autoconfigure</artifactId>
864
          <version>2.2.4.RELEASE</version>
865
866
        </dependency>
867
868
      3)pom.xml > right-click > Run As > Maven install
      [INFO] BUILD SUCCESS
869
870
871
872
    4. dependencyManagement 추가
873
      1)앞으로 추가되는 Library들의 Version을 일괄적으로 관리하기 위해 pom.xml에 Code 추가
874
      2)Mvnrepository에서 'spring boot dependencies'로 검색
875
      3)Spring Boot Dependencies에서 3.0.1.RELEASE
      4)아래의 Code를 pom.xml의 제일 아래에 추가
876
877
878
        <!--
        https://mvnrepository.com/artifact/org.apache.camel.springboot/camel-spring-boot-depe
        ndencies -->
879
        <dependency>
880
           <groupId>org.apache.camel.springboot</groupId>
           <artifactId>camel-spring-boot-dependencies</artifactId>
881
           <version>3.0.1</version>
882
883
           <scope>provided</scope>
```

```
884
         </dependency>
885
886
      5)pom.xml
887
         <dependencyManagement>
888
889
           <dependencies>
            <!--
890
            https://mvnrepository.com/artifact/org.apache.camel.springboot/camel-spring-boot-d
            ependencies -->
891
            <dependency>
892
               <groupId>org.apache.camel.springboot</groupId>
893
               <artifactId>camel-spring-boot-dependencies</artifactId>
894
               <version>3.0.1</version>
895
              <type>pom</type>
896
               <scope>provided</scope>
             </dependency>
897
898
           </dependencies>
899
         </dependencyManagement>
900
901
       6)project lombok library 추가
902
         <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
903
         <dependency>
904
           <groupId>org.projectlombok</groupId>
905
           <artifactId>lombok</artifactId>
           <version>1.18.12</version>
906
907
           <scope>provided</scope>
908
         </dependency>
909
910
       7)pom.xml > right-click > Run As > Maven install
911
        [INFO] BUILD SUCCESS
912
913
914
    5. 자동설정 구현하기
915
       1)이번 실습은 JDBC로 직접 Database 연동을 처리하는 Application을 위한 자동 설정이 목표이다.
       2)com.example.util package 생성
916
917
       3)com.example.util.JdbcConnectionManager.java 구현
918
919
        package com.example.util;
920
921
        import java.sql.Connection;
922
        import java.sql.DriverManager;
923
924
        import lombok. Setter;
925
        import lombok.ToString;
926
927
        @Setter
928
        @ToString
        public class JdbcConnectionManager {
929
930
           private String driverClass;
931
          private String url;
          private String username;
932
933
          private String password;
934
935
          public Connection getConnection() {
936
            try {
937
              Class.forName(this.driverClass);
938
              return DriverManager.getConnection(this.url, this.username, this.password);
939
            } catch (Exception e) {
```

```
940
              e.printStackTrace();
941
            }
942
            return null;
943
          }
        }
944
945
946
       4)JdbcConnectionManager를 bean으로 등록하는 환경 설정 Class를 작성한다.
947
       5)com.example.config package 생성
948
       6)com.example.config.UserAutoConfiguration.java 생성
949
950
        package com.example.config;
951
952
        import org.springframework.context.annotation.Bean;
953
        import org.springframework.context.annotation.Configuration;
954
955
        import com.example.util.JdbcConnectionManager;
956
957
        @Configuration
958
        public class UserAutoConfiguration {
959
          @Bean
960
          public JdbcConnectionManager getJdbcConnectionManager() {
961
            JdbcConnectionManager manager = new JdbcConnectionManager();
            manager.setDriverClass("oracle.jdbc.driver.OracleDriver");
962
963
            manager.setUrl("jdbc:oracle:thin:@localhost:1521:XE");
            manager.setUsername("hr");
964
965
            manager.setPassword("hr");
966
            return manager;
967
968
        }
969
970
971 6. src/main/resources folder 추가
972
       1)mybootstarter project > right-click > New > Source Folder
973
       2)Folder name: src/main/resources
974
       3)Finish
975
976
977
    7. resources/META-INF folder 생성
978
       1)src/main/resources > right-click > New > Folder
979
       2)Folder name: META-INF
980
      3)Finish
981
982
983 8. src/main/resources/META-INF/spring.factories file 생성
984
       org.springframework.boot.autoconfigure.EnableAutoConfiguration=\com.example.config.Us
      erAutoConfiguration
985
986
987 9. Build
988
       1)mybootstarter project > right-click > Run As > Maven install
989
        [INFO] BUILD SUCCESS
990
991
       2)성공하면 C:\Users\계정\.m2\repository\com\example\mybootstarter\0.0.1-SNAPSHOT에
      packaging한 jar 파일이 등록되어 있을 것이다.
992
993
994 10. Starter와 자동 설정 사용하기
995
       1)사용자 정의 Starter가 Maven Repository에 등록됐으면 이제 이 Starter를 이용하여 Application을 만들수
```

```
996
       2)위에서 생성한 Project의 pom.xml에서 아래의 Code를 복사한다.
         <groupId>com.example</groupId>
 997
         <artifactId>mybootstarter</artifactId>
 998
         <version>0.0.1-SNAPSHOT</version>
 999
1000
1001
       3)위의 Task4 에서 생성한 demo Project의 pom.xml의 <dependency>에 붙여넣는다.
1002
         <dependency>
1003
         <groupId>org.projectlombok</groupId>
1004
         <artifactId>lombok</artifactId>
1005
         </dependency>
1006
1007
         <!-- 아래 코드 추가 -->
1008
         <dependency>
           <groupId>com.example
1009
           <artifactId>mybootstarter</artifactId>
1010
1011
           <version>0.0.1-SNAPSHOT</version>
1012
         </dependency>
1013
1014
       4)pom.xml > right-click > Run As > Maven install
1015
         [INFO] BUILD SUCCESS
1016
1017
       5)위와 같이 BUILD SUCCESS가 나오면, demo Project의 Maven Dependencies의 목록에 보면
       mybootstarter가 추가된 것을 확인할 수 있다.
       6)이제 demo Project에 추가된 mybootstarter 를 사용하는 프로그래밍을 작성한다.
1018
       7)demo Project에 com.example.service package를 생성한다.
1019
1020
       8)com.example.service.JdbcConnectionManagerRunner Class를 생성한다.
1021
1022
         package com.example.service;
1023
1024
         import org.springframework.beans.factory.annotation.Autowired;
1025
         import org.springframework.boot.ApplicationArguments;
1026
         import org.springframework.boot.ApplicationRunner;
1027
         import org.springframework.stereotype.Service;
1028
         import com.example.util.JdbcConnectionManager;
1029
         @Service
1030
           public class JdbcConnectionManagerRunner implements ApplicationRunner {
1031
1032
1033
             @Autowired
1034
             private JdbcConnectionManager connectionManager;
1035
1036
             @Override
1037
             public void run(ApplicationArguments args) throws Exception {
               System.out.println("Connection Manager: " + this.connectionManager.toString());
1038
1039
             }
1040
           }
1041
1042
1043
       9)위에서 생선한 JdbcConnectionManagerRunner는 Container가 Component Scan하도록 @Service
       를 추가했다.
1044
       10)ApplicationRunner Interface를 구현했기 때문에 JdbcConnectionManagerRunner 객체가 생성되자
       마자 Container에 의해서 run() 가 자동으로 실행된다.
1045
       11)demo Project > right-click > Run As > Spring Boot App
1046
1047
       12)Console에 다음과 같은 출력이 나오면 자동설정이 정상적으로 동작했다는 의미이다.
1048
         Connection Manager: JdbcConnectionManager
         [driverClass=oracle.jdbc.driver.OracleDriver,
```

```
1049
          <u>url=jdbc:oracle:thin:@localhost:1521:XE</u>, username=hr, password=hr]
1050
1051
        13)만일 자동설정이 동작하지 않았다면 의존성 주입에서 Error가 발생했을 것이다
1052
1053
1054 11. 자동설정 재정의하기
1055
        1)Bean 재정의하기
1056
          -현재 mybootstarter는 Oracle과 Connection을 할 수 있다.
1057
          -이것을 MariaDB로 변경하려고 한다.
1058
          -demo Project에 com.example.config package 생성한다.
1059
          -com.example.config.UserConfiguration Class를 생성한다.
1060
1061
            package com.example.config;
1062
1063
           import org.springframework.context.annotation.Bean:
           import org.springframework.context.annotation.Configuration;
1064
           import com.example.util.JdbcConnectionManager;
1065
1066
1067
            @Configuration
1068
            public class UserConfiguration {
1069
1070
             @Bean
             public JdbcConnectionManager getJdbcConnectionManager() {
1071
1072
               JdbcConnectionManager manager = new JdbcConnectionManager();
               manager.setDriverClass("org.mariadb.jdbc.Driver");
1073
1074
               manager.setUrl("idbc:mariadb://localhost:3306/test");
               manager.setUsername("root");
1075
               manager.setPassword("javamariadb");
1076
1077
               return manager;
1078
             }
            }
1079
1080
1081
          -이렇게 하면 자동 설정으로 등록한 bean을 새로 등록한 bean이 덮어쓰면서 Oracle에서 MariaDB의
          JdbcConnectionManager를 사용할 수 있다.
1082
          -그런데, Application을 실행해 보면 Console에는 Error가 발생한다.
           The bean 'getJdbcConnectionManager', defined in class path resource
1083
           [com/example/config/UserAutoConfiguration.class], could not be registered. A bean
1084
           with that
1085
           name has already been defined in class path resource
1086
           [com/example/config/UserConfiguration.class] and overriding is disabled.
          -즉, Memory에 같은 Type의 bean이 두 개가 등록되어 충돌이 발행했다는 메시지이다.
1087
          -이 문제를 해결하기 위해서는 새로 생성된 bean이 기존에 등록된 bean을 덮어쓸 수 있도록 해야 한다.
1088
1089
          -demo Project의 src/main/resources/application.properties 파일에 다음의 설정을 추가한다.
1090
            ## Bean Overriding 설정
1091
           spring.main.allow-bean-definition-overriding=true
1092
1093
          *Properties Editor Plugin 설치하기
          1. application.properties 파일에 작성한 한글이 정상적으로 보이지 않으면 [Properties Editor] plugin을
1094
          섴
          치하면 된다.
1095
          2. Help > Install New Software... > Add...
1096
1097
          3. Name: Properties Editor
          4. Location: <a href="http://propedit.sourceforge.jp/eclipse/updates">http://propedit.sourceforge.jp/eclipse/updates</a>
1098
1099
          5. Add
1100
          6. 이렇게 설치하는 이유는 현재 Eclipse Marketplace에서 'Properties Editor'로 검색되지 않기 때문이다.
          7. 목록에서 [PropertiesEditor]만 Check하고 Next
1101
          8. 다른 Plugin 설치와 마찬가지로 계속 설치를 진행한다.
1102
1103
          9. 설치과정이 마치면 STS를 재 시작하고 application.properties file을 선택하고 Mouse right-click >
```

```
Open With > PropertiesEditor
1104
         10. 한글이 깨지지 않고 정상적으로 보이는 것을 볼 수 있다.
1105
1106
         -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
1107
         mybootstarter Project에서 등록한 JdbcConnectionManager를 사용했기 때문이다.
1108
         -이 문제를 해결하기 위한 Annotation이 바로 @Conditional이다.
         -@Conditional Annotation은 조건에 따라 새로운 객체를 생성할지 안할지를 결정할 수 있다.
1109
1110
1111
       2)@Conditional Annotation 사용하기
1112
         -@SpringBootApplication은 @EnableAutoConfiguration과 @ComponentScan을 포함하고 있다.
1113
         -Spring Boot는 @ComponentScan을 먼저 처리하여 사용자가 등록한 Bean을 먼저 Memory에 올린다.
1114
         -그리고 나중에 @EnableAutoConfiguration을 실행하여 자동 설정에 위한 Bean 등록을 처리한다.
1115
         -따라서 위에서 새로 생성한 Bean(MariaDB용 Connecitor)을 자동 설정한 Bean(Oracle Connector)이 덮
         어버린 것이다.
         -mybootstarter Project의 com.example.config.UserAutoConfiguration Class에
1116
         @ConditionalOnMissingBean Annotation을 적용한다.
1117
           package com.example.config;
1118
1119
1120
           import org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
1121
           import org.springframework.context.annotation.Bean;
1122
           import org.springframework.context.annotation.Configuration;
1123
           import com.example.util.JdbcConnectionManager;
1124
           @Configuration
1125
           public class UserAutoConfiguration {
1126
1127
             @Bean
1128
             @ConditionalOnMissingBean
             public JdbcConnectionManager getJdbcConnectionManager() {
1129
1130
              JdbcConnectionManager manager = new JdbcConnectionManager();
1131
              manager.setDriverClass("oracle.jdbc.driver.OracleDriver");
1132
              manager.setUrl("jdbc:oracle:thin:@localhost:1521:XE");
              manager.setUsername("hr");
1133
1134
              manager.setPassword("hr");
1135
              return manager;
1136
            }
           }
1137
1138
1139
         -@ConditionalONMissingBean은 등록하려는 Bean이 Memory에 없는 경우에만 현재의 Bean 등록을 처리
1140
         -따라서 사용자가 정의한 JdbcConnectionManager Bean이 @ComponentBean 설정에 의해 먼저 등록된
         다면 자동설정인 @EnableAutoConfiguration이 동작하는 시점에는 이미 등록된 Bean을 사용하고 새롭게
         Bean을 생성하지 않는다.
1141
         -이제 모두 저장하고 mybootstarter Project를 다시 install한다.
1142
           --mybootstarter Project > right-click > Run As > Maven install
1143
             [INFO] BUILD SUCCESS
         -그리고 demo Project를 Refresh하고 다시 Project를 실행하면 다음과 같이 Oracle에서 MariaDB로 변경된
1144
         것을 알 수 있다.
1145
           Connection Manager: JdbcConnectionManager [driverClass=org.mariadb.jdbc.Driver,
1146
           url=jdbc:mariadb://localhost:3306/test, username=root, password=javamariadb]
1147
       3)Property File 이용하기
1148
1149
         -Spring Container가 생성한 Bean의 Member Variable의 값이 자주 변경된다면 변경될 때마다 Java
         Source를 수정하기 보다 변경되는 정보만 Property로 등록하고 이 Property 정보를 이용해서 Bean을 생성하
         면 편리하다.
1150
         -Lab을 위해서 demo Project의 com.example.config.UserConfiguration.java의 @Configuration
         과
```

```
1151
1152
            @Bean을 주석처리한다.
1153
            //@Configuration
1154
            public class UserConfiguration {
1155
1156
              //@Bean
1157
              public JdbcConnectionManager getJdbcConnectionManager() {
1158
                JdbcConnectionManager manager = new JdbcConnectionManager();
1159
                manager.setDriverClass("org.mariadb.jdbc.Driver");
                manager.setUrl("jdbc:mariadb://localhost:3306/test");
1160
                manager.setUsername("root");
1161
                manager.setPassword("javamariadb");
1162
1163
                return manager;
1164
              }
            }
1165
1166
          -demo Project의 application.properites 파일에 다음 코드를 추가한다.
1167
1168
            ## Bean Overriding 설정
1169
            spring.main.allow-bean-definition-overriding=true
1170
            ## 데이터 소스: Oracle
1171
            user.jdbc.driverClass=oracle.jdbc.driver.OracleDriver
1172
            user.jdbc.url=jdbc:oracle:thin:@localhost:1521:XE
1173
            user.jdbc.username=hr
1174
            user.jdbc.password=hr
1175
1176
          -mybootstarter Project의 com.example.util.JdbcConnectionManagerProperties Class 추가로
          생성한다.
1177
          -이미 만들어 놓은 com.example.util.JdbcConnectionManager.java를 복사, 붙여넣기, 이름변경을
          JdbcConnectionManagerProeprties로 한다.
1178
1179
            package com.example.util;
1180
1181
            import java.sql.Connection;
            import java.sql.DriverManager;
1182
1183
            import org.springframework.boot.context.properties.ConfigurationProperties;
1184
            @ConfigurationProperties(prefix="user.jdbc")
1185
1186
            public class JdbcConnectionManagerProperties {
1187
              private String driverClass;
1188
              private String url:
1189
              private String username;
1190
              private String password;
1191
1192
              public String getDriverClass() {
1193
                return driverClass;
1194
1195
              public void setDriverClass(String driverClass) {
1196
                this.driverClass = driverClass;
1197
              }
1198
              public String getUrl() {
1199
                return url;
1200
1201
              public void setUrl(String url) {
1202
                this.url = url;
1203
              public String getUsername() {
1204
1205
                return username;
1206
              }
```

```
1207
              public void setUsername(String username) {
1208
               this.username = username;
1209
1210
              public String getPassword() {
1211
               return password;
1212
              public void setPassword(String password) {
1213
1214
               this.password = password;
1215
              }
1216
            }
1217
1218
          -위와 같이 작성하고 저장하면 노란색 경로라인이 발생한다.
1219
          -노란색 경고 메시지에 마우스를 올려놓으면 Add spring-boot-configuration-processotr to pom.xml
          link를 click한다.
          -이렇게 하면 자동으로 pom.xml에 다음과 같은 dependency가 추가된다.
1220
1221
            <dependency>
1222
              <groupId>org.springframework.boot</groupId>
1223
              <artifactId>spring-boot-configuration-processor</artifactId>
1224
              <optional>true</optional>
1225
            </dependency>
1226
1227
          -Error를 방지하기 위해 <version>2.2.0.RELEASE</version>을 추가한다.
1228
          -mybootstarter Project > right-click > Maven > Update Project > OK
1229
          -pom.xml > right-click > Run As > Maven install
1230
            [INFO] BUILD SUCCESS
          -이제 mybootstarter Project의 com.example.config.UserAutoConfiguration Class를 수정한다.
1231
1232
1233
            package com.example.config;
1234
1235
            import org.springframework.beans.factory.annotation.Autowired;
1236
            import org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
1237
            import org.springframework.boot.context.properties.EnableConfigurationProperties;
1238
            import org.springframework.context.annotation.Bean;
1239
            import org.springframework.context.annotation.Configuration;
1240
            import com.example.util.JdbcConnectionManager;
1241
            import com.example.util.JdbcConnectionManagerProperties;
1242
1243
            @Configuration
1244
            @EnableConfigurationProperties(JdbcConnectionManagerProperties.class)
1245
            public class UserAutoConfiguration {
1246
1247
              @Autowired
1248
              private JdbcConnectionManagerProperties properties;
1249
1250
              @Bean
1251
              @ConditionalOnMissingBean
1252
              public JdbcConnectionManager getJdbcConnectionManager() {
               JdbcConnectionManager manager = new JdbcConnectionManager();
1253
               manager.setDriverClass(this.properties.getDriverClass());
1254
1255
               manager.setUrl(this.properties.getUrl());
               manager.setUsername(this.properties.getUsername());
1256
1257
               manager.setPassword(this.properties.getPassword());
1258
               return manager;
1259
             }
1260
            }
1261
          -이제 mybootstarter Project를 다시 Install 한다.
1262
1263
            --mybootstarter Project > right-click > Run As > Maven install
```

```
1264
             [INFO] BUILD SUCCESS
         -demo Project를 Refresh하고 다시 실행한다.
1265
1266
           --다시 Oracle 설정 정보가 나오는 것을 알 수 있다.
1267
             Connection Manager: JdbcConnectionManager
             [driverClass=oracle.jdbc.driver.OracleDriver,
             url=jdbc:oracle:thin:@localhost:1521:XE, username=hr, password=hr]
1268
1269
         -만일 Database가 다시 MaraiDB로 변경된다면 demo Project의 application.properties를 다음과 같이
         변경하면 된다.
           ## Bean Overriding 설정
1270
1271
           spring.main.allow-bean-definition-overriding=true
1272
           ## 데이터 소스: Oracle
           #user.jdbc.driverClass=oracle.jdbc.driver.OracleDriver
1273
1274
           #user.jdbc.url=jdbc:oracle:thin:@localhost:1521:XE
           #user.jdbc.username=hr
1275
           #user.idbc.password=hr
1276
           ## 데이터 소스: MariaDB
1277
1278
           user.jdbc.driverClass=org.mariadb.jdbc.Driver
1279
           user.jdbc.url=jdbc:mariadb://localhost:3306/test
1280
           user.jdbc.username=root
1281
           user.jdbc.password=javamariadb
1282
1283
         -저장하면 바로 MaraiDB로 설정정보가 변경된 것을 알 수 있다.
1284
           Connection Manager: JdbcConnectionManager[driverClass=org.mariadb.jdbc.Driver,
1285
           url=jdbc:mariadb://localhost:3306/test, username=root, password=javamariadb]
1286
1287
1288
1289
1290 Task7. 간단한 JPA Project
1291 1. H2 Database 설치하기
1292
       1)여러 RDBMS가 있지만 H2를 사용하려는 이유는 Spring Boot가 기본적으로 H2를 지원하고 있기 때문이다.
1293
       2)H2는 Java로 만들어졌으며, 용량이 작고 실행 속도가 빠른 Open Source Database이다.
1294
       3)H2 Homepage(http://www.h2database.com/html/main.html)를 방문한다.
       4)Main page에서 Download의 All Platforms (zip, 8MB) Link를 Click하여 압축파일을 Download한다.
1295
1296
       5)h2-2019-10-14.zip 압축을 풀고 h2 Folder를 C:/Program Files로 이동한다.
       6)h2/bin의 h2w.bat를 실행하면 Browser기반의 관리 Console이 열린다.
1297
1298
       7)Tray에 있는 H2 Database Engine > right-click > Create a Database...을 실행하여 다음의 각 항목에
1299
       값을 입력하고 [Create] button을 click한다.
1300
         -Database path: ./test
1301
         -Username: sa
         -Password: javah2
1302
1303
         -Password confirmation: javah2
1304
         -Create button click
1305
1306
         Database was created successfully.
         JDBC URL for H2 Console:
1307
1308
         idbc:h2:./test
1309
1310
       8)각 항목의 정보를 입력하고 [Test Connection] 클릭해본다.
         -- Driver Class: org.h2.Driver
1311
1312
         --JDBC URL: jdbc:h2:~/test
1313
         --User Name: sa
1314
         -- Password: javah2
1315
1316
       9)연결이 성공하면 Web Console이 열린다.
1317
1318
```

```
2. JPA Project Installation
1320
        1)In STS, Help > Install New Software
        2) Work with: https://download.eclipse.org/releases/2019-12
1321
1322
1323
        4)결과에서
1324
          Web, XML, Java EE and OSGi Enterprise Development 하위의
1325
          -Dali Java Persistence Tools - EclipseLink JPA Support
1326
          -Dali Java Persistence Tools - JPA Diagram Editor
1327
          -Dali Java Persistence Tools - JPA Support
1328
          -m2e-wtp - JPA configurator for WTP (Optional)
1329
1330
        5)설치 후 STS Restart
1331
1332
1333 3. JPA Project 생성
        1)Package Explorer > right-click > Maven Project
1334
1335
1336
          -org.apache.maven.archetypes, maven-archetype-quickstart, 1.4
1337
          -Next
1338
1339
        2)각 항목 선택 후 Finish
1340
          -Group Id: com.example
1341
          -artifact Id: ipademo
1342
          -Version: 0.0.1-SNAPSHOT
1343
          -Package: com.example.jpademo
1344
          -Finish
1345
1346
        3)Project Facets 변환
          -jpademo Project > right-click > Properties > Project Facets
1347
1348
          -Java 1.8 > Runtimes > Apache Tomcat v9.0, jdk1.8.0 241 Check
1349
          -Check JPA
1350
          -만일 설정 화면 하단의 [Further configuration required...] Link에 Error message가 뜨는 경우
1351
            --Link click
1352
            --[JPA Facet] 창에서
1353
              ---Platform: Generic 2.1
              ---JPA implementation
1354
1355
                Type: Disable Library Configuration
1356
            --OK
          -Apply and Close
1357
1358
        4)Maven Project를 JPA Project로 변경하면 src/main/java하위에 META-INF/persistence.xml JPA 환
1359
        경설정 파일이 생긴다.
1360
1361
        5)jpademo Project의 Perspective를 JPA Perspective로 변경하려면
1362
          -Window > Perspective > Open Perspective > Other
1363
          -JPA 선택 > Open
1364
1365
1366 4. 의존성 추가
1367
        1)pom.xml을 수정
1368
          -Mvnrepository에서 'hibernate'로 검색하여 'Hibernate EntityManager Relocation'로 들어간다.
1369
          -5.4.12.Final Click
1370
1371
            <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager -->
1372
            <dependency>
1373
              <groupId>org.hibernate</groupId>
1374
              <artifactId>hibernate-entitymanager</artifactId>
1375
              <version>5.4.12.Final</version>
```

```
1376
           </dependency>
1377
1378
         -'h2'로 검색하여 'H2 Database Engine'으로 들어가서 1.4.200 선택
1379
           <!-- https://mvnrepository.com/artifact/com.h2database/h2 -->
1380
           <dependency>
             <groupId>com.h2database
1381
1382
             <artifactId>h2</artifactId>
1383
             <version>1.4.200</version>
1384
             <!--<scope>test</scope> --> 주의할 것, 이 scope tag는 반드시 삭제할 것
1385
           </dependency>
1386
         -'lombok'으로 검색하여
1387
1388
           <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
1389
           <dependency>
1390
              <groupId>org.projectlombok</groupId>
              <artifactId>lombok</artifactId>
1391
1392
              <version>1.18.12</version>
1393
              <scope>provided</scope>
1394
           </dependency>
1395
1396
         -pom.xml > right-click > Maven install
1397
           [INFO] BUILD SUCCESS
1398
1399
1400 5. Entity Class 작성 및 Table Mapping
1401
       1)Table을 준비한다.
1402
       2)JPA는 Table이 없으면 Java Class를 기준으로 Mapping할 Table을 자동으로 생성한다.
1403
       3)Table과 Mapping되는 Java Class를 Entity라고 한다.
1404
       4)JPA를 사용하는 데 있어서 가장 먼저 해야 할 일은 Entity를 생성하는 것이다.
1405
       5)Value Object Class처럼 Table과 동일한 이름을 사용하고 Column과 Mapping 될 Member Variable을
       선언하면 된다.
1406
       6)다만, Eclipse의 JPA Perspective가 제공하는 Entity 생성 기능을 사용하면 Entity를 생성함과 동시에 영속성
       설정 파일(persistence.xml)에 자동으로 Entity가 등록된다.
1407
       7)src/main/java Folder@ com.example.jpademo package > right-click > New > JPA Entity
1408
       8)다음 항목의 값을 입력 후 Finish 클릭
1409
         -Java package : com.example.domain
         -Class name: User
1410
1411
1412
       9)META-INF/persistence.xml 파일이 자동으로 수정되었다.
          <?xml version="1.0" encoding="UTF-8"?>
1413
          <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"</pre>
1414
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1415
1416
           xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
1417
           http://xmlns.jcp.org/xml/ns/persistence/persistence 2 1.xsd">
           <persistence-unit name="jpademo">
1418
1419
             <class>com.example.domain.User</class>
1420
           </persistence-unit>
1421
          </persistence>
1422
1423
       10)이제 방금 생성한 User Class를 수정한다.
1424
1425
         package com.example.domain;
1426
1427
         import java.io.Serializable;
1428
1429
         import javax.persistence.Entity;
1430
         import javax.persistence.Id;
1431
         import javax.persistence.Table;
```

```
1432
1433
          import lombok. Getter;
1434
          import lombok.Setter;
1435
          import lombok.ToString;
1436
         /**
1437
          * Entity implementation class for Entity: User
1438
1439
          */
1440
1441
          @Entity
1442
          @Table
1443
          @Getter
1444
          @Setter
1445
          @ToString
1446
          public class User implements Serializable {
1447
            @Id
1448
            private String userid;
1449
1450
            private String username;
1451
            private String gender;
1452
            private int age;
1453
            private String city;
1454
1455
            private static final long serialVersionUID = 1L;
1456
1457
            public User() {
1458
              super();
1459
1460
1461
          }
1462
1463
        11)다음은 JPA 사용하는 주요 Annotation을 설명한 것이다.
         -@Entity
1464
1465
            --Entity Class 임을 설명
1466
            --기본적으로 Class의 이름과 동일한 Table과 Mapping된다.
1467
1468
            --Entity의 이름과 Table의 이름이 다를 경우, name 속성을 이용하여 Mapping 한다.
            --이름이 동일하면 생략 가능
1469
1470
          -@Id
1471
            --Table의 primary key와 Mapping한다.
1472
            --Entiry의 필수 Annotation으로서 @Id가 없으면 Entity는 사용 불가
1473
          -@GeneratedValue
1474
            --@Id가 선언된 Field에 기본 키 값을 자동으로 할당
1475
1476
     6. JPA main 설정 파일 작성
1477
1478
        1)META-INF/persistence(JPA의 main 환경설정 파일) 수정
1479
          <?xml version="1.0" encoding="UTF-8"?>
          <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"</pre>
1480
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1481
            xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
1482
           http://xmlns.jcp.org/xml/ns/persistence/persistence 2 1.xsd">
1483
1484
            <persistence-unit name="jpademo">
1485
              <class>com.example.domain.User</class>
              cproperties>
1486
1487
                <!-- 필수 속성 -->
                cproperty name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>
1488
                cproperty name="javax.persistence.jdbc.user" value="sa"/>
1489
```

```
1490
                cproperty name="javax.persistence.jdbc.password" value="javah2"/>
                cproperty name="javax.persistence.jdbc.url" value="jdbc:h2:~/test"/>
1491
                cproperty name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
1492
1493
                <!-- Option 속성 -->
                property name="hibernate.show_sql" value="true"/>
1494
                property name="hibernate.format sql" value="true"/>
1495
                cproperty name="hibernate.use_sql_comments" value="false"/>
1496
                cproperty name="hibernate.id.new_generator_mappings" value="true"/>
1497
1498
                cproperty name="hibernate.hbm2ddl.auto" value="create"/>
1499
              </properties>
1500
            </persistence-unit>
1501
          </persistence>
1502
1503
        2)여기서 중요한 속성은 hibernate.dialect 이다.
1504
        3)이 속성은 JPA 구현체가 사용할 Dialect Class를 지정할 때 사용한다.
        4)이 속성을 H2Dialect Class로 설정하면 H2용 SQL이 생성되고, OracleDialect로 변경하면 Oracle용 SQL이
1505
        생성된다.
1506
1507
1508 7. JPA로 Data 처리하기
1509
        1)User 등록
1510
          -src/main/java Folder에 JPAClient Class를 생성한다.
1511
1512
            import javax.persistence.EntityManager;
1513
           import javax.persistence.EntityManagerFactory;
1514
            import javax.persistence.Persistence;
1515
1516
           import com.example.domain.User;
1517
1518
            public class JPAClient {
1519
             public static void main(String[] args) {
1520
               // EntityManager 생성
1521
               EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
1522
               EntityManager em = emf.createEntityManager();
1523
               try {
1524
                 User user = new User();
1525
                 user.setUserid("jimin");
                 user.setUsername("한지민");
1526
1527
                 user.setGender("a");
1528
                 user.setAge(24);
1529
                 user.setCity("서울");
1530
                 // 등록
1531
                 em.persist(user);
1532
               } catch (Exception e) {
1533
                 e.printStackTrace();
               } finally {
1534
1535
                 em.close();
1536
                 emf.close();
1537
               }
           }
1538
1539
1540
1541
          -JPA는 META-INF/persistence.xml을 먼저 loading한다.
1542
          -그리고 persistence.xml의 unit name으로 설정한 영속성 unit 정보를 이용하여 EntityManagerFactory
          객체를 생성한다.
1543
          -JPA를 이용하여 CRUD를 하려면 EntityManager 객체를 사용해야 한다.
1544
          -이것은 EntityManagerFactory를 통해 생성되며, EntityManager를 얻었으면 persist() 를 통해 User
          Table에 저장한다.
```

```
1545
1546
        2)실행하기
1547
          -JPAClient > right-click > Run As > Java Application
1548
          -만일 아래의 Error Message가 나오면
            ERROR: Database may be already in use: null. Possible solutions: close all other
1549
            connection(s); use the server mode [90020-200]
1550
          -작업관리자에서 javaw.exe 작업끝내기를 수행한다.
1551
1552
        3)실행 결과
1553
1554
          Hibernate:
1555
1556
          drop table User if exists
1557
1558
          Hibernate:
1559
1560
            drop sequence if exists hibernate_sequence
1561
          Hibernate: create sequence hibernate sequence start with 1 increment by 1
1562
1563
          Hibernate:
1564
            create table User (
1565
              userid varchar(255) not null,
1566
              age integer not null,
1567
              city varchar(255),
1568
              gender varchar(255),
1569
              username varchar(255),
1570
              primary key (userid)
1571
1572
1573
        4)하지만 실제 Database에는 Data가 Insert되지 않았다.
1574
1575
1576 8. Transaction 관리
        1)JPA가 실제 Table에 등록/수정/삭제 작업을 처리하기 위해서는 해당 작업이 반드시 Transaction안에서 수행되어
1577
        2)만약 Transaction을 시작하지 않았거나 등록/수정/삭제 작업 이후에 Transaction을 종료하지 않으면 요청한 작
1578
        업이 실제 Database에 반영되지 않는다.
1579
        3)JPAClient.java를 수정한다.
1580
1581
          import javax.persistence.EntityManager;
1582
          import javax.persistence.EntityManagerFactory;
1583
          import javax.persistence.EntityTransaction;
1584
          import javax.persistence.Persistence;
1585
1586
          import com.example.domain.User;
1587
1588
          public class JPAClient {
            public static void main(String[] args) {
1589
1590
              // EntityManager 생성
1591
              EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
1592
              EntityManager em = emf.createEntityManager();
             //Transaction 생성
1593
1594
              EntityTransaction tx = em.getTransaction();
1595
              try {
1596
               //Transaction 시작
1597
               tx.begin();
1598
1599
               User user = new User();
```

```
1600
               user.setUserid("jimin");
1601
               user.setUsername("한지민");
1602
               user.setGender("여");
1603
               user.setAge(24);
1604
               user.setCity("서울");
1605
               // 등록
1606
               em.persist(user);
1607
1608
               // Transaction commit
1609
               tx.commit();
1610
             } catch (Exception e) {
               e.printStackTrace();
1611
1612
               // Transaction rollback
1613
               tx.rollback();
1614
             } finally {
               em.close();
1615
1616
               emf.close();
1617
1618
           }
1619
         }
1620
1621
       4)실행하기
1622
         -JPAClient > right-click > Run As > Java Application
1623
1624
           Hibernate:
1625
             insert
1626
             into
1627
               User
1628
               (age, city, gender, username, userid)
1629
             values
1630
               (?,?,?,?,?)
1631
1632
        5)H2 Database Console에서 Run을 수행하면 방금 입력한 데이터가 삽입된 것을 볼 수 있다.
1633
1634
1635 9. 데이터 누적하기
1636
        1)현재 작성한 JPA 프로그램은 아무리 많이 실행해도 한 건의 Data만 등록된다.
1637
        2)즉 매번 Table이 새롭게 생성되기 때문이다.
1638
        3)따라서 JPA Client를 실행할 때마다 Data를 누적하기 위해서는 persistence.xml에서 다음을 수정해야 한다.
1639
1640
        cproperty name="hibernate.hbm2ddl.auto" value="create"/>
1641
1642
       4)다음으로 변경한다.
          cproperty name="hibernate.hbm2ddl.auto" value="update"/>
1643
1644
1645
       5)이렇게 변경하면 새롭게 생성하지 않고 기본의 Table을 재사용한다.
1646
        6)다음의 Data를 수행해서 3명의 User를 Insert한다.
1647
1648
         chulsu, 34, 부산, 남, 김철수
1649
         younghee, 44, 대전, 여, 이영희
1650
1651
1652
     10. Data 검색
1653
        1)JPAClient.java를 수정한다
1654
1655
         import javax.persistence.EntityManager;
1656
         import javax.persistence.EntityManagerFactory;
1657
         import javax.persistence.EntityTransaction;
```

```
1658
          import javax.persistence.Persistence;
1659
1660
          import com.example.domain.User;
1661
1662
          public class JPAClient {
            public static void main(String[] args) {
1663
1664
              // EntityManager 생성
1665
              EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
1666
              EntityManager em = emf.createEntityManager();
1667
              try {
                // User 검색
1668
                User user = em.find(User.class, "jimin");
1669
1670
                System.out.println("jimin --> " + user);
1671
              } catch (Exception e) {
1672
                e.printStackTrace();
1673
              } finally {
1674
                em.close();
1675
                emf.close();
1676
              }
1677
            }
1678
          }
1679
1680
        2)실행
1681
          Hibernate:
1682
1683
            select
1684
              user0_.userid as userid1_0_0_,
              user0_.age as age2_0_0_,
1685
1686
              user0_.city as city3_0_0_,
              user0 .gender as gender4 0 0 ,
1687
1688
              user0 .username as username5 0 0
1689
            from
1690
              User user0_
1691
            where
1692
              user0_.userid=?
1693
          jimin --> User [userid=jimin, username=한지민, gender=여, age=24, city=서울]
1694
1695
1696 11. Entity 수정
1697
        1)JPAClient.java 수정
1698
1699
          import javax.persistence.EntityManager;
1700
          import javax.persistence.EntityManagerFactory;
1701
          import javax.persistence.EntityTransaction;
1702
          import javax.persistence.Persistence;
1703
1704
          import com.example.domain.User;
1705
1706
          public class JPAClient {
1707
            public static void main(String[] args) {
1708
              // EntityManager 생성
1709
              EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
1710
              EntityManager em = emf.createEntityManager();
1711
              // Transaction 생성
1712
              EntityTransaction tx = em.getTransaction();
1713
              try {
                // Transaction 시작
1714
1715
                tx.begin();
```

```
1716
                // 수정할 User 조회
                User user = em.find(User.class, "younghee");
1717
1718
                user.setCity("광주");
1719
                user.setAge(55);
1720
                // Transaction commit
1721
                tx.commit();
1722
              } catch (Exception e) {
1723
                e.printStackTrace();
                // Transaction rollback
1724
1725
                tx.rollback();
1726
              } finally {
1727
                em.close();
1728
                emf.close();
1729
              }
1730
            }
          }
1731
1732
1733
        2)실행
1734
          Hibernate:
1735
            select
1736
              user0 .userid as userid1 0 0 ,
1737
              user0_.age as age2_0_0_,
1738
              user0_.city as city3_0_0_,
1739
              user0_.gender as gender4_0_0_,
1740
              user0_.username as username5_0_0_
1741
              User user0_
1742
1743
            where
1744
              user0_.userid=?
1745
          Hibernate:
1746
            update
1747
              User
1748
            set
1749
              age=?,
1750
              city=?,
1751
              gender=?,
1752
              username=?
1753
            where
1754
              userid=?
1755
1756
1757
      12. Entity 삭제
1758
        1)JPAClient.java 수정
1759
1760
          import javax.persistence.EntityManager;
          import javax.persistence.EntityManagerFactory;
1761
1762
          import javax.persistence.EntityTransaction;
1763
          import javax.persistence.Persistence;
1764
1765
          import com.example.domain.User;
1766
1767
          public class JPAClient {
1768
            public static void main(String[] args) {
1769
              // EntityManager 생성
1770
              EntityManagerFactory emf = Persistence.createEntityManagerFactory("ipademo");
1771
              EntityManager em = emf.createEntityManager();
              // Transaction 생성
1772
              EntityTransaction tx = em.getTransaction();
1773
```

```
1774
              try {
1775
                // Transaction 시작
1776
                tx.begin();
1777
1778
                // 삭제할 User 조회
1779
                User user = em.find(User.class, "younghee");
1780
                em.remove(user);
1781
                // Transaction commit
1782
1783
                tx.commit();
1784
              } catch (Exception e) {
                e.printStackTrace();
1785
1786
                // Transaction rollback
                tx.rollback();
1787
1788
              } finally {
1789
                em.close();
1790
                emf.close();
1791
1792
            }
1793
          }
1794
1795
        2)실행
          Hibernate:
1796
1797
            select
1798
              user0_.userid as userid1_0_0_,
              user0_.age as age2_0_0_,
1799
1800
              user0_.city as city3_0_0_,
              user0_.gender as gender4_0_0_,
1801
1802
              user0_.username as username5_0_0_
1803
            from
1804
              User user0
1805
            where
1806
              user0_.userid=?
          Hibernate:
1807
1808
            delete
1809
            from
1810
              User
1811
            where
1812
1813
1814 13. 여러 Record 조회와 JPQL
1815
        1)한 건의 Record 조회는 find()를 사용한다.
1816
        2)하지만, 여러 건의 Record를 조회하기 위해서는 JPQL(Java Persistence Query Language)라는 JPA에서
        제공하는 별도의 Query 명령어를 사용해야 한다.
1817
        3)JPAClient.java 수정
1818
1819
          import java.util.List;
1820
1821
          import javax.persistence.EntityManager;
1822
          import javax.persistence.EntityManagerFactory;
1823
          import javax.persistence.EntityTransaction;
1824
          import javax.persistence.Persistence;
1825
1826
          import com.example.domain.User;
1827
1828
          public class JPAClient {
1829
            public static void main(String[] args) {
1830
              // EntityManager 생성
```

```
1831
              EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
1832
              EntityManager em = emf.createEntityManager();
1833
1834
              // Transaction 생성
1835
              EntityTransaction tx = em.getTransaction();
1836
1837
              try {
                // Transaction 시작
1838
1839
                tx.begin();
1840
1841
                User user = new User();
                user.setUserid("hojune");
1842
1843
                user.setUsername("이호준");
1844
                user.setAge(30);
                user.setGender("남");
1845
                user.setCity("수원");
1846
1847
                // User 등록
1848
1849
                em.persist(user);
1850
1851
                // Transaction commit
1852
                tx.commit();
1853
                // 여러 Record 조회
1854
                String jpgl = "SELECT u FROM User u ORDER BY u.userid DESC";
1855
1856
                List<User> userList = em.createQuery(jpql, User.class).getResultList();
1857
                for (User usr : userList) {
                  System.out.println(usr);
1858
1859
1860
              } catch (Exception e) {
1861
                e.printStackTrace();
1862
                // Transaction rollback
1863
                tx.rollback();
              } finally {
1864
1865
                em.close();
                emf.close();
1866
1867
              }
1868
            }
1869
          }
1870
1871
        4)실행
          Hibernate:
1872
1873
            insert
1874
            into
1875
1876
              (age, city, gender, username, userid)
1877
            values
1878
              (?,?,?,?,?)
          Hibernate:
1879
1880
            select
1881
              user0_.userid as userid1_0_,
1882
              user0_.age as age2_0_,
1883
              user0_.city as city3_0_,
1884
              user0_.gender as gender4_0_,
              user0 .username as username5 0
1885
1886
            from
              User user0_
1887
1888
            order by
```

```
1889
             user0 .userid DESC
1890
          User [userid=mija, username=이미자, gender=여, age=60, city=대구]
1891
          User [userid=jimin, username=한지민, gender=여, age=24, city=서울]
1892
          User [userid=hojune, username=이호준, gender=남, age=30, city=수원]
          User [userid=chulsu, username=김철수, gender=남, age=34, city=부산]
1893
1894
1895
1896
1897
1898 Task8. 정적 Page 만들기
1899 1. Spring Boot project 생성
        1)Package Explorer > right-click > New > Spring Starter Project
1900
1901
        2)다음 각 항목의 값을 입력한 후, Next 클릭한다.
1902
          -Service URL: http://start.spring.io
1903
          -Name: springweb
1904
          -Type: Maven
1905
          -Packaging: jar
          -Java Version: 8
1906
1907
          -Language: Java
1908
         -Group: com.example
1909
          -Artifact : springweb
1910
         -Version: 0.0.1-SNAPSHOT
          -Description: Demo project for Spring Boot
1911
1912
          -Package: com.example.biz
1913
          -Next
1914
1915
        3)다음의 각 항목을 선택한 후 Finish 클릭
          -Spring Boot Version: 2.2.6
1916
1917
          -Developer Tools > Spring Boot DevTools
1918
          -Web > Spring Web
1919
1920
      2. Controller 생성
1921
1922
        1)src/main/java/com.example.biz > right-click > New > Class
1923
        2)Name: HomeController
1924
1925
          package com.example.biz;
1926
1927
          import org.springframework.stereotype.Controller;
1928
          import org.springframework.web.bind.annotation.GetMapping;
1929
          import org.springframework.web.servlet.ModelAndView;
1930
1931
          @Controller
1932
         public class HomeController {
            @GetMapping("/")
1933
            public ModelAndView home(ModelAndView mav) {
1934
1935
             mav.setViewName("index.html");
1936
             return mav;
1937
1938
1939
1940
1941 3. static file 생성
1942
        1)src/main/resources/static/images folder 생성
1943
          -spring-boot.png 추가할 것
1944
1945
        2)src/main/resources/static/js folder 생성
1946
          -jquery-3.5.0.min.js 추가할 것
```

```
1947
1948
       3)src/main/resources/static/css folder 생성
1949
         -bootstrap.min.css 추가할 것
1950
1951
       4)src/main/resources/static/index.html
1952
         <!DOCTYPE html>
1953
1954
         <html>
1955
         <head>
1956
         <meta charset="UTF-8">
1957
         <title>Home page</title>
         <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css" />
1958
1959
         <script src="js/jquery-3.5.0.min.js"></script>
1960
         <script>
1961
           $(document).ready(function() {
1962
             alert("Hello, Spring Boot World!!!");
1963
1964
         </script>
1965
         </head>
1966
         <body>
1967
           <div>
1968
             <img src="images/spring-boot.png"/>
1969
           </div>
1970
           <div class="jumbotron">
             <h1>Hello, Spring Boot World</h1>
1971
1972
             ...
1973
             >
1974
               <a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>
1975
             1976
           </div>
1977
         </body>
1978
         </html>
1979
1980 4. Spring Boot에서는 template을 사용하지 않을 경우 기본적으로 static file은 src/main/resources/static에
     서 찾는다.
1981 5. 이럴 때는 반드시 file의 확장자 .html까지 넣어야 한다.
1982 6. springweb Project > right-click > Run As > Spring Boot App
1983 7. <a href="http://localhost:8080/">http://localhost:8080/</a>
1984
1985
1986
1987 -----
1988 Task9. JSP Page 만들기
1989 1. Spring Boot에서는 JSP 사용을 권장하지 않는다.
1990
     2. 'jar' 형식으로 동작하지 않고 War file로 배포해야 하는 등의 몇 가지 제약이 있어서이기도 하지만, 가장 큰 이유는 이
     미 JSP 자체가 Server 측 언어로 그 사용 빈도가 줄고 있기 때문이다.
     3. view 부분에 code가 섞여서 logic을 분리하기 어렵고, HTML과 같은 tag를 사용하므로 HTML 편집기 등에서 JSP
     삽입 부분을 분리하기 어려우며 Visual 편집기 등에서도 사용이 어렵다.
1992 4. 그래서 template을 통해 code를 분리해야 할 필요가 있는 것이다.
1993
1994
     5. Spring Boot project 생성
1995
       1)Package Explorer > right-click > New > Spring Starter Project
1996
1997
       2)다음 각 항목의 값을 입력 후 Next 클릭
1998
         -Service URL :http://start.spring.io
1999
         -Name: springispdemo
2000
         -Type: Maven
2001
         -Packaging : jar
```

```
2002
          -Java Version: 8
2003
          -Language: Java
2004
          -Group: com.example
2005
          -Artifact: springjspdemo
2006
          -Version: 0.0.1-SNAPSHOT
2007
          -Description: Demo project for Spring Boot
2008
          -Package: com.example.biz
2009
        3)각 항목 선택 후 Finish 클릭
2010
2011
          -Spring Boot Version: 2.2.6
          -Web > Spring Web > Finish
2012
2013
2014
2015 6. pom.xml
2016
        1)jstl을 위해
2017
          <dependency>
2018
            <groupId>javax.servlet</groupId>
2019
            <artifactId>jstl</artifactId>
2020
            <version>1.2</version>
2021
          </dependency>
2022
2023
       2)JSP를 위해
2024
          <dependency>
2025
            <groupId>org.apache.tomcat
2026
            <artifactId>tomcat-jasper</artifactId>
2027
            <version>9.0.33</version>
2028
          </dependency>
2029
2030
        3)pom.xml > right-click > Run As > Maven install
2031
          [INFO] BUILD SUCCESS
2032
2033
2034 7. folder 준비
2035
        1)src/main folder 안에 webapp folder 생성
2036
        2)webapp folder 안에 WEB-INF folder 생성
        3)WEB-INF folder 안에 jsp folder 생성
2037
2038
2039 8. 만일 JSP를 template으로 사용하는 경우에는 이 WEB-INF folder 안에 template file을 준비할 필요가 있다.
2040
2041 9. src/main/resources/application.properties code 추가
2042
        1)application.properties > right-click > Open with > Generic Editor
2043
          spring.mvc.view.prefix : /WEB-INF/jsp/
2044
          spring.mvc.view.suffix:.jsp
2045
2046
     10. jsp folder 안에 jsp file 생성
2047
2048
        1)index.jsp
2049
          <%@ page language="java" contentType="text/html; charset=UTF-8"</pre>
2050
          pageEncoding="UTF-8"%>
2051
          <@@ page import="java.util.Date, java.text.SimpleDateFormat" %>
2052
          <!DOCTYPE html>
2053
2054
          <html>
2055
            <head>
2056
              <meta charset="UTF-8">
2057
              <title>Insert title here</title>
2058
            </head>
            <body>
2059
```

```
2060
             <h1>Index page</h1>
2061
             <%=new SimpleDateFormat("yyyy년 MM월 dd일").format(new Date()) %>
2062
            </body>
2063
          </html>
2064
2065
2066 11. HomeController class 생성
        1)com.example.biz > right-click > New > Click
2067
        2)Name: HomeController
2068
2069
        3)Finish
2070
2071
          package com.example.biz;
2072
2073
          import org.springframework.stereotype.Controller;
2074
          import org.springframework.web.bind.annotation.GetMapping;
2075
2076
          @Controller
2077
          public class HomeController {
2078
2079
            @GetMapping("/")
2080
           public String index() {
2081
             return "index";
2082
2083
          }
2084
2085
2086 12. 실행
       http://localhost:8080/
2087
2088
2089
          Index page
          2020년 04월 20일
2090
2091
2092
2093
2094
2095 Task10. thymeleaf template 사용하기
2096 1. Spring Boot project 생성
        1)Package Explorer > right-click > New > Spring Starter Project
2097
2098
        2)다음의 각 항목 입력 후 Next 클릭
2099
          -Service URL: http://start.spring.io
2100
          -Name: MyBootWeb
          -Type: Maven
2101
2102
          -Packaging: jar
2103
          -Java Version: 8
2104
          -Language: Java
2105
          -Group: com.example
          -Artifact: MyBootWeb
2106
          -Version: 0.0.1-SNAPSHOT
2107
2108
          -Description: Demo project for Spring Boot
2109
          -Package: com.example.biz
2110
2111
       3)각 항목 선택 후 Finish 클릭
2112
          -Spring Boot Version: 2.2.6
2113
          -Web > Spring Web > Finish
2114
2115
2116 2. src/main/java/com.example.biz.MyBootWebApplication.java
2117
```

```
2118
        package com.example.biz;
2119
2120
        import org.springframework.boot.SpringApplication;
2121
        import org.springframework.boot.autoconfigure.SpringBootApplication;
2122
2123
        @SpringBootApplication
2124
        public class MyBootWebApplication {
2125
          public static void main(String[] args) {
2126
            SpringApplication.run(MyBootWebApplication.class, args);
2127
2128
        }
2129
2130
2131 3. 실행
2132
        1)MyBootWebApplication.java > right-click > Run As > Spring Boot App
          -http://localhost:8080/
2133
2134
2135
          Whitelabel Error Page
2136
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
          There was an unexpected error (type=Not Found, status=404).
2140
2141
          No message available
2142
2143
2144 4. Controller 작성하기
        1)com.example.biz > right-click > New > Class
2145
2146
        2)Name: HelloController > Finish
2147
        3)HelloController.java
2148
2149
          package com.example.biz;
2150
2151
          import org.springframework.web.bind.annotation.RequestMapping;
2152
          import org.springframework.web.bind.annotation.RestController;
2153
2154
          @RestController
2155
          public class HelloController {
2156
            @RequestMapping("/")
2157
2158
            public String index() {
2159
              return "Hello Spring Boot World!";
2160
            }
2161
          }
2162
2163
2164 5. project > right-click > Run As > Spring Boot App
2165
2166 6. http://localhost:8080/
2167
2168
        Hello Spring Boot World!
2169
2170
2171 7. 매개변수 전달
2172
        1)HelloController.java 수정
2173
2174
          @RequestMapping("/{num}")
2175
          public String index(@PathVariable int num) {
```

```
2176
            int result = 0;
2177
            for(int i = 1; i \le num; i++) result += i;
            return "total: " + result;
2178
2179
          }
2180
2181
2182 8. project > right-click > Run As > Spring Boot App
2183
2184 9. http://localhost:8080/100
2185
2186
        total: 5050
2187
2188
2189 10. pom.xml에 lombok dependency 추가
2190
2191
        <dependency>
2192
         <groupId>org.projectlombok</groupId>
2193
         <artifactId>lombok</artifactId>
2194
        <version>1.18.12</version>
2195
        <scope>provided</scope>
2196
        </dependency>
2197
2198
        -pom.xml > right-click > Run As > Maven install
2199
          [INFO] BUILD SUCCESS
2200
2201
2202
      11. 객체를 JSON으로 출력하기
2203
        1)src/main/java/com.example.biz/Student.java 생성
2204
2205
          package com.example.biz;
2206
2207
          import lombok.AllArgsConstructor;
2208
          import lombok.Data;
2209
2210
          @Data
          @AllArgsConstructor
2211
2212
          public class Student {
2213
            private int userid;
2214
            private String name;
2215
            private int age;
2216
            private String address;
2217
2218
2219
        2)HelloController.java 수정
2220
2221
          @RestController
2222
          public class HelloController {
            String [] names = {"조용필", "이미자", "설운도"};
2223
            int [] ages = \{56, 60, 70\};
2224
            String [] addresses = {"서울특별시", "부산광역시", "대전광역시"};
2225
2226
2227
            @RequestMapping("/{userid}")
            public Student index(@PathVariable int userid) {
2228
2229
              return new Student(userid, names[userid], ages[userid], addresses[userid]);
2230
2231
          }
2232
        3)http://localhost:8080/1
2233
```

```
{"userid":1,"name":"이미자","age":60,"address":"부산광역시"}
2234
2235
2236
2237
      11. thymeleaf 추가하기
2238
        1)pom.xml 수정하기
2239
          -Select Dependencies tab > Add
2240
          -Group Id: org.springframework.boot
2241
          -Artifact Id: spring-boot-starter-thymeleaf
2242
          -Version:
2243
          -Scope: compile
2244
          -OK
2245
2246
          <dependency>
2247
            <groupId>org.springframework.boot</groupId>
2248
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
2249
          </dependency>
2250
2251
        2)HelloController.java 수정
2252
2253
          package com.example.biz;
2254
2255
          import org.springframework.stereotype.Controller;
2256
          import org.springframework.web.bind.annotation.RequestMapping;
2257
2258
          @Controller
          public class HelloController {
2259
2260
2261
            @RequestMapping("/")
2262
            public String index() {
2263
              return "index";
2264
2265
          }
2266
2267
        3)template file 생성
2268
          -src/main/resources/templates > right-click > New > Other...> Web > HTML File > Next
2269
          -File name: index.html
2270
2271
          <!doctype html>
2272
          <html lang="en">
2273
            <head>
              <meta charset="UTF-8" /> <!--반드시 종결 tag 필요 -->
2274
2275
              <title>Index Page</title>
2276
              <style type="text/css">
2277
                h1 { font-size:18pt; font-weight:bold; color:gray; }
2278
                body { font-size:13pt; color:gray; margin:5px 25px; }
2279
              </style>
2280
            </head>
2281
            <body>
2282
              <h1>Hello! Spring Boot with Thymeleaf</h1>
2283
              This is sample web page.
2284
            </body>
2285
          </html>
2286
2287
        4)http://localhost:8080/
2288
2289
          Hello! Spring Boot with Thymeleaf
2290
          This is sample web page
2291
```

```
2292
        5)template에 값 표시하기
2293
          -index.html code 수정
2294
2295
          <body>
2296
            <h1>Hello! Spring Boot with Thymeleaf</h1>
2297
            2298
          </body>
2299
          -HelloController.java 수정
2300
2301
            @Controller
2302
           public class HelloController {
2303
             @RequestMapping("/{num}")
2304
             public String index(@PathVariable int num, Model model) {
2305
2306
               int result = 0:
               for(int i = 1; i \le num; i++) result += i;
2307
               model.addAttribute("msg", "total : " + result);
2308
2309
               return "index";
2310
             }
2311
            }
2312
2313
        6)http://localhost:8080/100
2314
2315
          Hello! Spring Boot with Thymeleaf
         total: 5050
2316
2317
2318
        7)ModelAndView class 사용하기
2319
          -HelloController.java 수정
2320
2321
          @Controller
2322
          public class HelloController {
2323
            @RequestMapping("/{num}")
2324
           public ModelAndView index(@PathVariable int num, ModelAndView mav) {
2325
             int result = 0;
2326
             for(int i = 1; i \le num; i++) result += i;
             mav.addObject("msg", "total : " + result);
2327
2328
             mav.setViewName("index");
2329
             return mav;
2330
2331
          }
2332
2333
        8)http://localhost:8080/100
2334
2335
          Hello! Spring Boot with Thymeleaf
         total: 5050
2336
2337
2338
        9)form 사용하기
2339
          -index.html 수정
2340
2341
            <!doctype html>
2342
            <html lang="en">
2343
             <head>
                <meta charset="UTF-8"/>
2344
2345
                <title>Index Page</title>
2346
                <style type="text/css">
                 h1 { font-size:18pt; font-weight:bold; color:gray; }
2347
                 body { font-size:13pt; color:gray; margin:5px 25px; }
2348
2349
                </style>
```

```
2350
             </head>
2351
             <body>
               <h1>Hello!</h1>
2352
2353
               <form method="post" action="/send">
2354
                 <input type="text" name="txtName" th:value="${value}" />
2355
2356
                 <input type="submit" value="Send" />
               </form>
2357
             </body>
2358
2359
           </html>
2360
2361
         -HelloController.java 수정
2362
2363
           @Controller
2364
           public class HelloController {
2365
             @RequestMapping(value = "/", method = RequestMethod.GET)
2366
             public ModelAndView index(ModelAndView mav) {
2367
2368
               mav.setViewName("index");
2369
               mav.addObject("msg", "Please write your name...");
2370
               return mav;
2371
             }
2372
2373
             @RequestMapping(value = "/send", method = RequestMethod.POST)
             public ModelAndView send(@RequestParam("txtName") String name, ModelAndView
2374
               mav.addObject("msg", "안녕하세요!" + name + "님!");
2375
               mav.addObject("value", name);
2376
               mav.setViewName("index");
2377
2378
               return mav;
2379
             }
2380
           }
2381
2382
        10)http://localhost:8080
2383
        11)기타 form controller
2384
         -index.html 수정
2385
2386
2387
         <!doctype html>
2388
          <html lang="en">
2389
           <head>
             <meta charset="UTF-8" />
2390
2391
             <title>Index Page</title>
2392
             <style type="text/css">
2393
               h1 {
2394
                 font-size: 18pt;
2395
                 font-weight: bold;
2396
                 color: gray;
2397
2398
               body {
2399
                 font-size: 13pt;
2400
                 color: gray;
                 margin: 5px 25px;
2401
2402
               }
2403
             </style>
2404
           </head>
2405
           <body>
           <h1>Hello!</h1>
2406
```

```
2407
           Please wait...
           <form method="post" action="/">
2408
2409
           <div>
             <input type="checkbox" id="ckeck1" name="check1" /> <label for="ckeck1">체크
2410
             </label>
           </div>
2411
2412
           <div>
             <input type="radio" id="male" name="gender" value="male" /> <label for="male">
2413
             남성</label>
2414
           </div>
           <div>
2415
             <input type="radio" id="female" name="gender" value="female" /> <label
2416
             for="female">여성</label>
           </div>
2417
2418
           <div>
             <select name="selOs" size="4">
2419
2420
               <option>--선택--</option>
               <option value="Windows">windows</option>
2421
2422
               <option value="MacOS">MacOS</option>
2423
               <option value="Linux">Linux</option>
2424
             </select>
2425
             <select name="selEditors" size="4" multiple="multiple">
2426
               <option>--선택--</option>
2427
               <option value="Notepad">Notepad</option>
2428
               <option value="Editplus">Editplus
2429
               <option value="Visual Studio Code">Visual Studio Code</option>
               <option value="Sublime Text">Sublime Text
2430
               <option value="Eclipse">Eclipse</option>
2431
2432
             </select>
2433
           </div>
           <input type="submit" value="전송" />
2434
2435
           </form>
         </body>
2436
2437
        </html>
2438
2439
        12)HelloController.java 수정
2440
2441
         @Controller
2442
         public class HelloController {
2443
2444
           @RequestMapping(value = "/", method = RequestMethod.GET)
           public ModelAndView index(ModelAndView mav) {
2445
2446
2447
             mav.setViewName("index");
             mav.addObject("msg", "값을 입력후 전송버튼을 눌러주세요.");
2448
2449
             return mav;
           }
2450
2451
           @RequestMapping(value = "/", method = RequestMethod.POST)
2452
           public ModelAndView send(@RequestParam(value="check1", required=false) boolean
2453
           check1,
2454
                 @RequestParam(value="gender", required=false) String gender,
                 @RequestParam(value="selOs", required=false) String selOs,
2455
2456
                 @RequestParam(value="selEditors", required=false) String [] selEditors,
                 ModelAndView mav) {
             String result = "";
2457
2458
             try {
2459
               result = "check: " + check1 + ", gender: " + gender + ", OS: " + selOs +
```

```
"\nEditors: ";
             }catch(NullPointerException ex) {}
2460
2461
2462
             try {
2463
               result += selEditors[0];
               for(int i = 1; i < selEditors.length; i++) result += ", " + selEditors[i];
2464
2465
             }catch(NullPointerException ex) {
               result += "null";
2466
2467
             }
2468
             mav.addObject("msg", result);
2469
2470
             mav.setViewName("index");
2471
             return mav;
2472
           }
          }
2473
2474
2475
        13)http://localhost:8080
2476
2477
2478 12. Redirect
2479
        1)index.html 수정
2480
2481
          <body>
2482
            <h1>Hello! index.</h1>
2483
          </body>
2484
2485
        2)HelloController.java 수정
2486
2487
          @Controller
          public class HelloController {
2488
2489
2490
            @RequestMapping("/")
2491
           public ModelAndView index(ModelAndView mav) {
             mav.setViewName("index");
2492
2493
             return mav;
2494
            }
2495
2496
            @RequestMapping("/other")
2497
           public String other() {
2498
             return "redirect:/";
2499
2500
2501
            @RequestMapping("/home")
           public String home() {
2502
2503
             return "forward:/";
2504
2505
         }
2506
        3)redirect와 forward 차이점 구분하기
2507
2508
2509
2510
2511 -----
2512 Task11. thymeleaf template 사용하기2
2513 1. Spring Boot project 생성
2514
        1)Package Explorer > right-click > New > Spring Starter Project
2515
2516
       2)다음 각 항목의 값을 입력 후 Next 클릭
```

```
2517
          -Service URL : <a href="http://start.spring.io">http://start.spring.io</a>
2518
          -Name: templatedemo
2519
          -Type: Maven
2520
          -Packaging: Jar
          -Java Version: 8
2521
2522
          -Language: Java
2523
          -Group: com.example
2524
          -Artifact: templatedemo
2525
          -Version: 0.0.1-SNAPSHOT
2526
          -Description: Demo project for Spring Boot
2527
          -Package: com.example.biz
2528
2529
       3)다음 각 항목 선택 후 Finish 클릭
2530
          -Spring Boot Version: 2.2.6
          -Select
2531
2532
           --Developer Tools > Spring Boot DevTools, Lombok
2533
           --Web > Spring Web
2534
            --Template Engines > Thymeleaf
2535
          -Finish
2536
2537
2538
     2. HomeController.java 생성
2539
        1)com.example.biz > right-click > New > Class
2540
2541
        2)Name: HomeController
2542
2543
          package com.example.biz;
2544
2545
          import org.springframework.stereotype.Controller;
2546
          import org.springframework.web.bind.annotation.GetMapping;
2547
2548
          @Controller
2549
          public class HomeController {
2550
            @GetMapping("/")
2551
2552
            public String home() {
2553
             return "index";
2554
2555
          }
2556
2557
        3)index.html 생성
2558
          -src/main/resources/templates > New > Other > Web > HTML File > Next
2559
          -File name: index.html
2560
          <!DOCTYPE html>
2561
          <html>
2562
2563
            <head>
             <meta charset="UTF-8" />
2564
2565
              <title>Insert title here</title>
2566
            </head>
            <body>
2567
2568
             <h1>Hello Page</h1>
2569
              2570
            </body>
2571
          </html>
2572
2573
       4)실행
          --http://localhost:8080
2574
```

```
2576
        Hello Page
2577
        Fri Feb 21 00:31:37 KST 2020
2578
2579
2580 3. Utility Object 사용하기
2581
      1)index.html 수정
2582
2583
      <body>
2584
        <h1>Hello Page</h1>
2585
        2586
2587
        2588
      </body>
2589
2590
      2)실행
2591
        Hello Page
2592
2593
        09/11/2019 00:15
2594
2595
        0001234
2596
2597
        WELCOME TO SPRING
2598
2599
2600 4. 매개변수에 접근하기
2601
      1)index.html의 <body> 부분을 아래와 같이 수정한다.
2602
2603
        <body>
          <h1>Hello page</h1>
2604
          2605
          param.name[0]">
2606
        </body>
2607
2608
      2)HomeController.java를 수정한다.
2609
2610
        @RequestMapping("/")
2611
        public ModelAndView index(ModelAndView mav) {
          mav.setViewName("index");
2612
2613
          return may:
2614
        }
2615
2616
        @RequestMapping("/home")
2617
        public String home() {
         return "forward:/";
2618
2619
2620
2621
      3)그리고 id와 name을 query string으로 지정해서 접속한다.
        -http://localhost:8080/home/?id=javaexpert&name=Springboot
2622
2623
        -결과는 아래와 같다.
2624
          Helo page
2625
         from parameter... id=javaexpert,name=Springboot
2626
2627
      4)controller를 거치지 않고 template내에서 직접 전달된 값을 사용할 수 있다.
2628
      5)중요한 것은 param내의 id나 name 배열에서 첫 번째 요소를 지정해서 추출한다는 것이다.
      6)그 이유는 query string으로 값을 전송할 때 같은 이름의 값을 여러 개 전송하기 위해서다.
2629
2630
      7)예를 들면, <a href="http://localhost:8080/home/?id=123&id=456&name=javaexpert&name=peter">http://localhost:8080/home/?id=123&id=456&name=javaexpert&name=peter</a>
2631
      8)이렇게 하면 param에서 추출하는 id와 name이 각각 {123,456}, {javaexpert,peter}가 되는 것이기 때문
```

```
이다.
2632
        9)여기서 사용하고 있는 th:text의 값을 보면 큰따옴표 안에 다시 작은 따옴표를 사용해서 값을 작성하고 있다.
2633
        10)이것은 OGNL로 text literal을 작성할 때 사용하는 방식이다.
2634
        11)이렇게 하면 다수의 literal을 연결할 때 큰 따옴표안에 작은 따옴표 literal을 사용할 수 있게 된다.
2635
2636
          th:text="one two three"
          th:text="'one' + 'two ' + 'three'"
2637
2638
2639
2640 5. Message식 사용하기
2641
        1)src/main/resources > right-click > New > File
2642
        2)File name: messages.properties > Finish
2643
2644
          content.title=Message sample page.
2645
          content.message=This is sample message from properties.
2646
2647
        3)index.html 수정
2648
2649
          <body>
2650
          <h1 th:text="#{content.title}">Hello page</h1>
2651
          2652
          </body>
2653
2654
       4)실행
2655
          Message sample page.
2656
          This is sample message from properties.
2657
2658
2659 6. Link식과 href
2660
        1)index.html
2661
2662
          <body>
2663
            <h1 th:text="#{content.title}">Helo page</h1>
2664
             < a th:href="@{/home/{orderId}(orderId=${param.id[0]})}">link</a>
2665
          </body>
2666
2667
        2)접속할 때 query string에 id를 지정한다.
2668
        3)예를 들어 <a href="http://localhost:8080/?id=123에 접속하면 link에는 /home/123이 설정된다.">http://localhost:8080/?id=123에 접속하면 link에는 /home/123이 설정된다.</a>
2669
2670
2671 7. 선택 객체와 변수식
2672
        1)src/main/java/com.example.biz > right-click > New > Class
2673
        2)Name: Member
2674
2675
          package com.example.biz;
2676
2677
          import lombok.AllArgsConstructor;
2678
          import lombok.Data;
2679
2680
          @Data
2681
          @AllArgsConstructor
2682
          public class Member {
2683
            private int id;
           private String username;
2684
2685
           private int age;
          }
2686
2687
2688
        3)HomeController.java 수정
```

```
2689
2690
        @RequestMapping("/")
2691
        public ModelAndView index(ModelAndView mav) {
2692
          mav.setViewName("index");
          mav.addObject("msg","Current data.");
2693
          Member obj = new Member(123, "javaexpert",24);
2694
2695
          mav.addObject("object",obj);
2696
          return mav;
2697
        }
2698
       4)index.html 수정
2699
2700
2701
        <!DOCTYPE HTML>
2702
        <html xmlns:th="http://www.thymeleaf.org">
2703
2704
            <title>top page</title>
2705
            <meta charset="UTF-8" />
2706
            <stvle>
             h1 { font-size:18pt; font-weight:bold; color:gray; }
2707
2708
             body { font-size:13pt; color:gray; margin:5px 25px; }
2709
             tr { margin:5px; }
2710
             th { padding:5px; color:white; background:darkgray; }
             td { padding:5px; color:black; background:#e0e0ff; }
2711
2712
            </style>
2713
          </head>
2714
          <body>
2715
            <h1 th:text="#{content.title}">Hello page</h1>
2716
            message.
2717
            ID
2718
             NAME
2719
2720
              AGE
2721
            2722
          </body>
        </html>
2723
2724
2725
       5)실행
2726
2727
       6)controller에서 object를 저장해둔 Member 값이 표 형태로 출력된다
2728
2729
2730
2731 --
2732 Task12. Spring Boot와 JDBC 연동하기
2733
     1. Spring Boot project 생성
2734
       1)Package Explorer > right-click > New > Spring Starter Project
2735
       2)다음의 각 항목의 값을 입력후 Next 클릭
2736
       -Service URL: http://start.spring.io
      -Name: BootJdbcDemo
2737
       -Type: Maven
2738
2739
       -Packaging: Jar
      -Java Version: 8
2740
2741
      -Language: Java
2742
       -Group: com.example
2743
      -Artifact: BootJdbcDemo
2744
       -Version: 0.0.1-SNAPSHOT
      -Description: Demo project for Spring Boot
2745
2746
      -Package: com.example.biz
```

```
2747
2748
       3)다음 각 항목을 선택후 Finish 클릭
2749
         -Spring Boot Version: 2.2.6
2750
         -Select
2751
           --SQL > check MySQL, JDBC API
2752
           --Web > Spring Web
2753
           --Developer Tools > Spring Boot DevTools, Lombok
2754
         -Finish
2755
2756
       4)위에서 type을 선택시 고려사항
         -만일 Embeded된 tomcat으로 Stand-Alone형태로 구동시키기 위한 목적이라면 Packaging Type을 jar로
2757
         선택한다.
2758
         -war로 선택할 경우, 기존과 같이 외부의 tomcat으로 deploy 하는 구조로 만들어진다.
2759
         -물론, source의 최종배포의 형태가 server의 tomcat에 deploy해야 하는 구조라면 처음부터 war로 만들어서
         작업해도 상관없다.
         -jar로 선택하고, local에서 개발 및 test를 하다가 나중에 배포할 경우 war로 변경해서 배포를 할 수도 있다.
2760
2761
2762
2763 2. pom.xml
2764
       1)jstl 사용을 위한
2765
         <dependency>
2766
           <groupId>javax.servlet</groupId>
2767
           <artifactId>istl</artifactId>
2768
           <version>1.2</version>
         </dependency>
2769
2770
2771
       2)jasper 사용을 위한
2772
         <dependency>
2773
           <groupId>org.apache.tomcat</groupId>
2774
           <artifactId>tomcat-jasper</artifactId>
2775
           <version>9.0.33</version>
2776
         </dependency>
2777
2778
       3)MariaDB 사용을 위한
2779
         <dependency>
           <groupId>org.mariadb.jdbc</groupId>
2780
2781
           <artifactId>mariadb-java-client</artifactId>
2782
           <version>2.5.4</version>
2783
         </dependency>
2784
2785
2786
       4)pom.xml > right-click > Run As > Maven install
2787
       [INFO] BUILD SUCCESS
2788
2789
2790 3. src/main/resources/application.properties
2791
       spring.application.name=BootJDBCDemo
       spring.datasource.url=idbc:mariadb://localhost:3306/test
2792
2793
       spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
2794
       spring.datasource.username=root
2795
       spring.datasource.password=javamariadb
2796
2797
2798 4. Table 생성
2799
       CREATE TABLE test. User
2800
         username VARCHAR(20) PRIMARY KEY,
2801
         age TINYINT NOT NULL
2802
```

```
2803
        );
2804
2805
2806 5. VO object 생성
2807
        1)src/main/java > right-click > New > Package
2808
        2)Name: com.example.vo
2809
        3)com.example.vo > right-click > New > Class
2810
        4)Name: UserVO
2811
2812
          package com.example.vo;
2813
2814
          import lombok.Data;
2815
          import lombok.AllArgsConstructor;
2816
          import lombok.NoArqsConstructor;
2817
2818
          @Data
2819
          @AllArgsConstructor
2820
          @NoArgsConstructor
2821
          public class UserVO {
2822
            private String username;
2823
            private int age;
2824
          }
2825
2826
2827 6. Dao object 생성
2828
        1)src/main/java > right-click > New > Package
2829
        2)Name: com.example.dao
2830
        3)com.example.dao > right-click > New > Inteface
2831
        4)Name: UserDao
2832
2833
          package com.example.dao;
2834
2835
          import java.util.List;
2836
2837
          import com.example.vo.UserVO;
2838
2839
          public interface UserDao {
            int create(UserVO userVO);
2840
2841
            List<UserVO> readAll();
2842
            UserVO read(String username);
2843
            int update(UserVO userVO);
            int delete(String username);
2844
2845
          }
2846
2847
        5)com.example.dao > right-click > New > Class
        6)Name: UserDaoImpl
2848
2849
          package com.example.dao;
2850
2851
2852
          import java.sql.ResultSet;
2853
          import java.sql.SQLException;
2854
          import java.util.List;
2855
2856
          import org.springframework.beans.factory.annotation.Autowired;
2857
          import org.springframework.jdbc.core.JdbcTemplate;
          import org.springframework.jdbc.core.RowMapper;
2858
          import org.springframework.stereotype.Repository;
2859
2860
```

```
import com.example.vo.UserVO;
2862
2863
          @Repository
          public class UserDaoImpl implements UserDao {
2864
2865
2866
            @Autowired
            private JdbcTemplate jdbcTemplate;
2867
2868
            class UserMapper implements RowMapper<UserVO> {
2869
              public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2870
2871
                UserVO user = new UserVO();
2872
                user.setUsername(rs.getString("username"));
2873
                user.setAge(rs.getInt("age"));
2874
                return user;
2875
2876
2877
            @Override
2878
            public int create(UserVO userVO) {
2879
2880
              String sql = "INSERT INTO User(username, age) VALUES(?, ?)";
2881
              return this.jdbcTemplate.update(sql, userVO.getUsername(), userVO.getAge());
2882
            }
2883
2884
            @Override
2885
            public List<UserVO> readAll() {
2886
              String sql = "SELECT * FROM User";
2887
              return this.jdbcTemplate.query(sql, new UserMapper());
2888
2889
            @Override
2890
2891
            public UserVO read(String username) {
              String sql = "SELECT * FROM User WHERE username = ?";
2892
              return this.jdbcTemplate.queryForObject(sql, new Object[] {username}, new
2893
              UserMapper());
2894
            }
2895
2896
            @Override
2897
            public int update(UserVO userVO) {
              String sql = "UPDATE User SET age = ? WHERE username = ?";
2898
              return this.jdbcTemplate.update(sql, userVO.qetAqe(), userVO.qetUsername());
2899
2900
            }
2901
            @Override
2902
2903
            public int delete(String username) {
2904
              String sql = "DELETE FROM User WHERE username = ?";
2905
              return this.jdbcTemplate.update(sql, username);
2906
            }
2907
2908
          }
2909
2910
2911 7. Controller
2912
        1)com.example.biz > right-click > New > Class
2913
        2)Name: MainController
2914
2915
          package com.example.biz;
2916
2917
          import org.springframework.beans.factory.annotation.Autowired;
```

```
2918
          import org.springframework.stereotype.Controller;
2919
          import org.springframework.ui.Model;
2920
          import org.springframework.web.bind.annotation.GetMapping;
2921
          import org.springframework.web.bind.annotation.PostMapping;
2922
2923
          import com.example.dao.UserDao;
2924
          import com.example.vo.UserVO;
2925
          @Controller
2926
2927
          public class MainController {
2928
            @Autowired
2929
            private UserDao userDao;
2930
2931
            @GetMapping("/")
            public String index() {
2932
2933
              return "index";
2934
2935
2936
            @PostMapping("/user")
2937
            public String insert(UserVO userVO) {
2938
              this.userDao.create(userVO);
2939
              return "redirect:/user";
2940
2941
            @GetMapping("/user")
2942
2943
            public String list(Model model) {
              model.addAttribute("users", this.userDao.readAll());
2944
2945
              return "list";
2946
            }
2947
          }
2948
2949
2950 8. static resources 준비
2951
        1)src/main/resources/static/images folder 생성
2952
          -spring-boot.png 추가할 것
2953
2954
        2)src/main/resources/static/js folder 생성
2955
          -jquery-3.4.1.min.js 추가할 것
2956
2957
        3)src/main/resources/static/css folder 생성
2958
          -style.css
2959
2960
          @charset "UTF-8";
2961
          body {
2962
            background-color:yellow;
2963
          h1{
2964
2965
           color : blue;
2966
2967
2968
2969 9. JSP를 위한 folder 준비
2970
        1)기본적으로 Spring Boot 에서는 jsp파일을 인식이 되지 않는다.
2971
        2)그래서 만일 jar로 packaging 한다면 embedded tomcat이 인식하는 web루트를 생성한다.
2972
        3)src > main 폴더 밑에 webapp와 jsp가 위치할 폴더를 만들어준다.
2973
        4)src/main folder 안에 webapp folder 생성
2974
        5)webapp folder 안에 WEB-INF folder 생성
2975
        6)WEB-INF folder 안에 jsp folder 생성
```

```
2976
        7)만일 JSP를 template으로 사용하는 경우에는 이 WEB-INF folder 안에 template file을 준비할 필요가 있다.
        8)folder를 다 만들었으면, WEB 루트(Context Root)로 인식할 있도록 환경설정을 아래와 같이 추가한다.
2977
2978
         -src/main/resources/application.properties code 추가
2979
2980
           spring.mvc.view.prefix: /WEB-INF/jsp/
2981
           spring.mvc.view.suffix: .jsp
2982
2983
2984 10. jsp folder 안에 jsp file 생성
2985
        1)index.jsp
2986
          <@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>
2987
          <@@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
         <!DOCTYPE html>
2988
2989
         <html>
2990
           <head>
             <meta charset="UTF-8">
2991
2992
             <title>New User Insertion Page</title>
2993
             <link rel="stylesheet" type="text/css" href="/css/style.css">
2994
             <c:url var="jsurl" value="/js/jquery-3.3.1.slim.min.js" />
2995
             <script src="${jsurl}"></script>
2996
             <script>
               $(document).ready(function() {
2997
                 alert("Hello, Spring Boot!");
2998
2999
               });
3000
             </script>
3001
           </head>
3002
           <body>
3003
             <img src="/images/spring-boot.png"/>
3004
             <h1>New User Insertion Page</h1>
             <form action="/user" method="post">
3005
               Name : <input type="text" name="username" /><br />
3006
               Age: <input type="number" name="age" /><br />
3007
               <button type="submit">Submit</button>
3008
3009
             </form>
           </body>
3010
          </html>
3011
3012
3013
        2)list.isp
        <@ page language="java" contentType="text/html; charset=UTF-8"
3014
        pageEncoding="UTF-8"%>
        <@@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3015
3016
        <!DOCTYPE html>
3017
        <html>
         <head>
3018
           <meta charset="UTF-8">
3019
3020
           <title>User List</title>
           <link rel="stylesheet" type="text/css" href="/css/style.css">
3021
3022
         </head>
3023
          <body>
           <h1>User List</h1>
3024
3025
           <c:forEach var="user" items="${users}">
3026
3027
               ${user.username}(${user.age})
3028
             </c:forEach>
3029
           </body>
3030
        </html>
3031
```

```
3032
3033
3034 11. src/main/java/com.example.biz/BootJdbcDemoApplication.java
3035
3036
       package com.example.biz;
3037
3038
       import org.springframework.boot.SpringApplication;
3039
       import org.springframework.boot.autoconfigure.SpringBootApplication;
3040
       import org.springframework.context.annotation.ComponentScan;
3041
3042
       @SpringBootApplication
       @ComponentScan("com.example")
3043
                                         <-- 추가 code
3044
       public class BootJdbcDemoApplication {
3045
3046
         public static void main(String[] args) {
3047
           SpringApplication.run(BootJdbcDemoApplication.class, args);
3048
       }
3049
3050
3051
       <u>-@SpringBootConfiguration</u> 은 다음의 annotation 3개를 모두 담고 있다.
3052
         --@SpringBootConfiguration
3053
         --@EnableAutoConfiguration
3054
          --@ComponentScan
3055
       -만약, AutoScan이 되어야 하는 component class들 - 대표적으로 @Controller, @Service,
       @Repository, @Component등-의 위치가 main class가 위치한 package보다 상위 package에 있거나, 하
       위가 아닌 다른 package에 있는 경우, scan이 되지 않는다.
3056
       -Controller class가 main의 하위 class에 있으면 상관없지만, 예를 들어, main class가 다른 package나 하위
       package가 아니면 아래와 같이 해줘야 한다.
3057
       -명시적으로 ComponentScan을 할 Base Package를 지정해주면 된다.
3058
         --ex) @ComponentScan(basePackages = "com.springboot.demo")
3059
3060 14)project > right-click > Run As > Spring Boot App
     15)http://localhost:8080
3061
3062
3063
3064
3065 -----
3066 Task13. Spring Boot와 JPA 연동하기
3067
       1. Spring Boot의 Database 처리는 기본적으로 JPA 기술을 기반으로 하고 있다.
3068
3069
       2. 이 JPA를 Spring Framework에서 사용할 수 있게 한 것이 'Spring Data JPA' framework이다.
3070
3071
       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 사용
       할 수 있다.
3072
3073
       4. 즉, 이 library는 각종 library를 조합해서 간단히 database 접속을 구현하게 한 기능이다.
3074
3075
       5. Spring Boot project 생성
3076
         1)Package Explorer > right-click > New > Spring Starter Project
3077
         2)다음 각 항목의 값을 입력한 후 Next 클릭
3078
           -Service URL : http://start.spring.io
3079
           -Name: JpaDemo
3080
           -Type: Maven
3081
           -Packaging: Jar
           -Java Version: 8
3082
3083
           -Language: Java
3084
           -Group: com.example
```

```
3085
            -Artifact: JpaDemo
3086
            -Version: 0.0.1-SNAPSHOT
3087
            -Description: Demo project for Spring Boot
3088
            -Package: com.example.biz
3089
          3)다음 각 항목을 선택한 후 Finish 클릭
3090
3091
            -Spring Boot Version: 2.2.4
3092
            -Select
              --SQL > Spring Data JPA, H2 Database
3093
3094
              --Developer Tools > Spring Boot DevTools
3095
            -Finish
3096
3097
3098 6. Entity
3099
        1)JPA에서 Entity라는 것은 Database에 저장하기 위해서 정의한 class이다.
3100
        2)일반적으로 RDBMS에서 Table 같은 것이다.
3101
        3)com.example.biz > right-click > New > Class
3102
3103
        4)Name: MemberVO
3104
3105
          package com.example.biz;
3106
          import javax.persistence.Column;
3107
3108
          import javax.persistence.Entity;
          import javax.persistence.GeneratedValue;
3109
3110
          import javax.persistence.GenerationType;
3111
          import javax.persistence.Id;
3112
3113
3114
          @Entity(name="Member")
          public class MemberVO {
3115
3116
            @GeneratedValue(strategy = GenerationType.AUTO)
3117
            private long id;
3118
3119
            @Column
3120
            private String username;
3121
            @Column
3122
            private int age;
3123
3124
            public MemberVO() {}
3125
            public MemberVO(String username, int age) {
3126
3127
              this.username = username;
3128
              this.age = age;
3129
3130
3131
            public long getId() {
3132
              return id;
3133
3134
3135
            public void setId(long id) {
3136
              this.id = id;
3137
3138
3139
            public String getUsername() {
3140
              return username;
3141
3142
```

```
3143
           public void setUsername(String username) {
             this.username = username;
3144
3145
3146
3147
           public int getAge() {
3148
             return age;
3149
3150
           public void setAge(int age) {
3151
3152
             this.age = age;
3153
3154
3155
           @Override
           public String toString() {
3156
             return "MemberVO [id=" + id + ", username=" + username + ", age=" + age + "]";
3157
3158
3159
3160
3161
       5)여기서 주의할 점은 기본 생성자는 반드시 넣어야 한다.
3162
3163
3164
     7. Repository
       1)Entity class를 구성했다면 이번엔 Repository interface를 만들어야 한다.
3165
3166
       2)Spring Framework에서는 Entity의 기본적인 삽입, 조회, 수정, 삭제가 가능하도록 CrudRepository라는
       inteface가 있다.
3167
       3)com.example.biz > right-click > New > Interface
3168
3169
       4) Name: MemberRepository
3170
3171
         package com.example.biz;
3172
3173
         import java.util.List;
3174
3175
         import org.springframework.data.jpa.repository.Query;
         import org.springframework.data.repository.CrudRepository;
3176
3177
         import org.springframework.data.repository.query.Param;
3178
3179
         public interface MemberRepository extends CrudRepository<MemberVO, Long> {
3180
           List<MemberVO> findByUsernameAndAgeLessThan(String username, int age);
3181
3182
           @Query("select t from Member t where username= :username and age < :age")</pre>
           List < Member VO > find By Username And Age Less Than SQL (@Param ("username") String
3183
           username, @Param("age") int age);
3184
3185
           List<MemberVO> findByUsernameAndAgeLessThanOrderByAgeDesc(String username,
           int age);
         }
3186
3187
3188
         -위의 코드는 실제로 MemberVO Entity를 이용하기 위한 Repository class이다.
3189
         -기본적인 method 외에도 추가적인 method를 지정할 수 있다.
3190
         -method 이름을 기반(Query Method)으로 해서 만들어도 되고 @Query를 이용해 기존의 SQL처럼 만들어도
         된다.
3191
         -findByUsernameAndAgeLessThan method와 findByUsernameAndAgeLessThanSQL method
         는 같은 결과를 출력하지만 전자의 method는 method 이름을 기반으로 한 것이고 후자의 method는 @Query
         annotation을 기반으로 해서 만든 것이다.
         -method 이름 기반으로 해서 만들면 추후에 사용할 때 method 이름만으로도 어떤 query인지 알 수 있다는 장
3192
         점이 있다.
         -반대로 @Query annotation으로 만든 method는 기존의 source를 converting하는 경우 유용하게 사용할
3193
```

```
수 있다.
3194
          -@Query
            --다만 @Query annotation으로 query를 만들 때에 from 절에 들어가는 table은 Entity로 지정된 class
3195
3196
          -method 이름 기반 작성법
3197
          -해당 부분은 Spring 문서
          (<a href="https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.guery-methods">https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.guery-methods</a>.
          query-creation)를 통해 확인할 수 있다.
3198
3199
3200 8. Application 작성
3201
        1)Entity 및 Repository가 준비 되었다면 실제로 @SpringBootApplication annotation이 있는 class에서
        실제로 사용해 보자.
3202
3203
          package com.example.biz;
3204
3205
          import java.util.List;
3206
3207
          import org.springframework.beans.factory.annotation.Autowired;
3208
          import org.springframework.boot.CommandLineRunner;
          import org.springframework.boot.SpringApplication;
3209
3210
          import org.springframework.boot.autoconfigure.SpringBootApplication;
3211
          @SpringBootApplication
3212
3213
          public class JpaDemoApplication implements CommandLineRunner {
3214
            @Autowired
3215
            MemberRepository memberRepository;
3216
3217
            public static void main(String[] args) {
3218
              SpringApplication.run(JpaDemoApplication.class, args);
3219
3220
            @Override
3221
3222
            public void run(String... args) throws Exception {
              memberRepository.save(new MemberVO("a", 10));
3223
              memberRepository.save(new MemberVO("b", 15));
3224
              memberRepository.save(new MemberVO("c", 10));
3225
              memberRepository.save(new MemberVO("a", 5));
3226
3227
              Iterable<MemberVO> list1 = memberRepository.findAll();
3228
              System.out.println("findAll() Method.");
3229
              for (MemberVO m : list1) {
3230
                System.out.println(m.toString());
3231
3232
              System.out.println("findByUserNameAndAgeLessThan() Method.");
3233
              List<MemberVO> list2 = memberRepository.findByUsernameAndAgeLessThan("a",
              for (MemberVO m : list2) {
3234
3235
                System.out.println(m.toString());
3236
              }
              System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
3237
3238
              List<MemberVO> list3 =
              memberRepository.findByUsernameAndAgeLessThanSQL("a", 10);
3239
              for (MemberVO m : list3) {
3240
                System.out.println(m.toString());
3241
              }
3242
              System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
3243
              List<MemberVO> list4 =
3244
```

```
memberRepository.findByUsernameAndAgeLessThanOrderByAgeDesc("a", 15);
             for (MemberVO m : list4) {
3245
3246
               System.out.println(m.toString());
3247
             }
3248
             memberRepository.deleteAll();
3249
3250
3251
         }
3252
3253
3254 9. 실행
3255
3256
       findAll() Method.
3257
       MemberVO [id=1, username=a, age=10]
3258
       MemberVO [id=2, username=b, age=15]
       MemberVO [id=3, username=c, age=10]
3259
3260
       MemberVO [id=4, username=a, age=5]
3261
       findByUserNameAndAgeLessThan() Method.
3262
       MemberVO [id=4, username=a, age=5]
3263
       findByUserNameAndAgeLessThanSQL() Method.
3264
       MemberVO [id=4, username=a, age=5]
3265
       findByUserNameAndAgeLessThanSQL() Method.
       MemberVO [id=1, username=a, age=10]
3266
3267
       MemberVO [id=4, username=a, age=5]
3268
3269
3270 -----
3271 Task14. Spring Boot JPA
3272
       1. Spring Boot project 생성
3273
         1)Package Explorer > right-click > New > Spring Starter Project
3274
3275
         2)다음의 각 항목의 값을 입력 후 Next 클릭
3276
           -Service URL :http://start.spring.io
           -Name : BootJpaDemo
3277
3278
           -Type: Maven
3279
           -Packaging: Jar
           -Java Version: 8
3280
3281
           -Language: Java
3282
           -Group: com.example
3283
           -Artifact : BootJpaDemo
3284
           -Version: 0.0.1-SNAPSHOT
3285
           -Description: Demo project for Spring Boot
3286
           -Package: com.example.biz
3287
3288
         3)각 항목을 선택 후 Finish 클릭
           -Spring Boot Version: 2.2.4
3289
3290
           -Select
             --SQL > Spring Data JPA, H2 Database
3291
             --Developer Tools > Spring Boot DevTools
3292
             --Web > Spring Web
3293
3294
             --Template Engines > Thymeleaf
3295
           -Finish
3296
3297
         4)DevTools
3298
           -It provides developer tools.
           -These tools are helpful in application development mode.
3299
           -One of the features of developer tool is automatic restart of the server for any change
3300
           in code
```

```
3301
3302
3303 2. Entity 작성하기
        1)com.example.biz > right-click > New > Package
3304
3305
        2)Name: com.example.biz.vo
3306
        3)com.example.biz.vo > right-click > New > Class
3307
        4)Name: User
3308
          package com.example.biz.vo;
3309
3310
3311
          import javax.persistence.Column;
3312
          import javax.persistence.Entity;
3313
          import javax.persistence.GeneratedValue;
3314
          import javax.persistence.GenerationType;
3315
          import javax.persistence.Id;
3316
          import javax.persistence.Table;
3317
3318
          import lombok.Data;
3319
3320
          @Entity
3321
          @Table
3322
          public class User {
3323
3324
            DI@
3325
            @GeneratedValue(strategy = GenerationType.AUTO)
3326
            @Column
3327
            private long id;
3328
3329
            @Column(length = 20, nullable = false)
3330
            private String username;
3331
3332
            @Column(length = 100, nullable = true)
3333
            private String email;
3334
3335
            @Column(nullable = true)
3336
            private Integer age;
3337
3338
            @Column(nullable = true)
3339
            private String memo;
3340
3341
            public long getId() {
3342
              return id;
3343
3344
3345
            public void setId(long id) {
              this.id = id;
3346
3347
            }
3348
3349
            public String getUsername() {
3350
              return username;
3351
3352
3353
            public void setUsername(String username) {
3354
              this.username = username;
3355
3356
            public String getEmail() {
3357
              return email;
3358
```

```
3359
            }
3360
3361
            public void setEmail(String email) {
             this.email = email;
3362
3363
3364
            public Integer getAge() {
3365
3366
             return age;
3367
3368
3369
            public void setAge(Integer age) {
3370
             this.age = age;
3371
            }
3372
3373
            public String getMemo() {
3374
              return memo;
3375
3376
            public void setMemo(String memo) {
3377
3378
             this.memo = memo;
3379
            }
3380
          }
3381
3382
3383 3. Repository 생성하기
3384
        1)com.example.biz > right-click > New > Package
3385
        2)Name: com.example.biz.dao
        3)com.example.biz.dao > right-click > New > Interface
3386
3387
        4)Name: UserRepository > Finish
3388
3389
          package com.example.biz.dao;
3390
3391
          import org.springframework.data.jpa.repository.JpaRepository;
          import org.springframework.stereotype.Repository;
3392
3393
3394
          import com.example.biz.vo.User;
3395
3396
          @Repository("repository")
3397
          public interface UserRepository extends JpaRepository < User, Long > {
3398
3399
          }
3400
          -JpaRepository라는 interface는 새로운 repository를 생성하기 위한 토대가 된다.
3401
          -모든 Repository는 이 JpaRepository를 상속해서 작성한다
3402
3403
3404
3405 4. HelloController 작성
3406
        1)com.example.biz > right-click > New > Class
3407
        2)Name: HelloController
3408
3409
          package com.example.biz;
3410
3411
          import org.springframework.beans.factory.annotation.Autowired;
3412
          import org.springframework.stereotype.Controller;
3413
          import org.springframework.web.bind.annotation.RequestMapping;
3414
          import org.springframework.web.servlet.ModelAndView;
3415
3416
          import com.example.biz.dao.UserRepository;
```

```
3417
         import com.example.biz.vo.User;
3418
3419
         @Controller
3420
         public class HelloController {
3421
3422
           @Autowired
           UserRepository repository;
3423
3424
           @RequestMapping("/")
3425
           public ModelAndView index(ModelAndView mav) {
3426
             mav.setViewName("index");
3427
             mav.addObject("msg", "this is sample content.");
3428
3429
             Iterable<User> list = repository.findAll();
             mav.addObject("data", list);
3430
3431
             return mav;
3432
         }
3433
3434
3435
         -UserRepository에는 findAll 같은 method가 정의되어 있지 않다.
3436
         -이것은 부모 interface인 JpaRepository가 가지고 있는 method이다.
         -이를 통해 모든 entity가 자동으로 추출되는 것이다.
3437
3438
3439
3440 5. JUnit Test
3441
        1)src/test/java/com.example.biz.BootJpaDemoApplicationTests.java > right-click > Run As
        > JUnit Test
3442
        2)Green bar
3443
3444
3445 6. template 준비하기
3446
        1)src/main/resources > right-click > New > File
3447
3448
        2) File name: messages.properties
3449
         content.title=Message sample page.
3450
         content.message=This is sample message from properties.
3451
        3)src/main/resources/templates > right-click > New > Web > HTML Files
3452
3453
       4)Name: index.html
3454
          <!DOCTYPE HTML>
3455
          <html xmlns:th="http://www.thymeleaf.org">
3456
           <head>
3457
             <title>top page</title>
3458
             <meta charset="UTF-8" />
3459
3460
             <style>
               h1 { font-size:18pt; font-weight:bold; color:gray; }
3461
               body { font-size:13pt; color:gray; margin:5px 25px; }
3462
3463
               pre { border: solid 3px #ddd; padding: 10px; }
3464
             </style>
3465
           </head>
3466
           <body>
             <h1 th:text="#{content.title}">Hello page</h1>
3467
3468
             3469
           </body>
          </html>
3470
3471
3472
3473 7. 실행해서 접속
```

```
3474
       1)저장돼있는 data가 회색 사각 틀 안에 표시된다.
3475
       2)아직 아무 data가 없기 때문에 빈 배열 []라고 표시된다.
3476
3477
3478 8. Entity의 CRUD 처리하기: form으로 data 저장하기
3479
       1)index.html 수정
3480
3481
         <!DOCTYPE HTML>
         <html xmlns:th="http://www.thymeleaf.org">
3482
3483
         <head>
3484
         <title>top page</title>
3485
         <meta charset="UTF-8" />
3486
         <style>
3487
        h1 {
3488
          font-size: 18pt;
3489
          font-weight: bold;
3490
          color: gray;
3491
         }
3492
3493
        body {
3494
          font-size: 13pt;
3495
          color: gray;
3496
          margin: 5px 25px;
3497
        }
3498
3499
        tr {
3500
          margin: 5px;
3501
3502
        th {
3503
3504
          padding: 5px;
3505
          color: white;
          background: darkgray;
3506
         }
3507
3508
3509
        td {
3510
          padding: 5px;
3511
          color: black;
3512
          background: #e0e0ff;
3513
         }
3514
         </style>
3515
         </head>
3516
         <body>
          <h1 th:text="#{content.title}">Hello page</h1>
3517
3518
          <form method="post" action="/" th:object="${formModel}">
3519
3520
              <label for="username">이름</label>
3521
                <input type="text" name="username" th:value="*{username}" />
3522
3523
              3524
              3525
                <label for="age">연령</label>
3526
                <input type="text" name="age" th:value="*{age}" />
3527
              3528
              <label for="email">메일</label>
3529
3530
                <input type="text" name="email" th:value="*{email}" />
3531
```

```
3532
3533
               <label for="memo">메모</label>
               <textarea name="memo" th:text="*{memo}" cols="20"
3534
               rows="5"></textarea>
3535
             3536
             3537
               3538
               /
3539
             3540
            </form>
3541
          3542
          <hr />
3543
          3544
            3545
             ID
3546
             이름
3547
            3548
3549
             3550
             3551
            3552
          3553
        </body>
3554
        </html
3555
3556
       2)HelloController.java 수정
3557
3558
        package com.example.biz;
3559
3560
        import org.springframework.beans.factory.annotation.Autowired;
3561
        import org.springframework.stereotype.Controller;
3562
        import org.springframework.transaction.annotation.Transactional;
3563
        import org.springframework.web.bind.annotation.ModelAttribute;
3564
        import org.springframework.web.bind.annotation.RequestMapping;
3565
        import org.springframework.web.bind.annotation.ReguestMethod;
        import org.springframework.web.servlet.ModelAndView;
3566
3567
3568
        import com.example.biz.dao.UserRepository;
3569
        import com.example.biz.vo.User;
3570
3571
        @Controller
        public class HelloController {
3572
3573
3574
          @Autowired
3575
          UserRepository repository;
3576
          @RequestMapping(value = "/", method = RequestMethod.GET)
3577
          public ModelAndView index(@ModelAttribute("formModel") User mydata, ModelAndView
3578
          mav) {
3579
            mav.setViewName("index");
            mav.addObject("msg", "this is sample content.");
3580
3581
           Iterable<User> list = repository.findAll();
            mav.addObject("datalist", list);
3582
3583
            return mav;
3584
          }
3585
          @RequestMapping(value = "/", method = RequestMethod.POST)
3586
          @Transactional(readOnly = false)
3587
```

```
3588
           public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
           mav) {
3589
             repository.saveAndFlush(mydata);
3590
             return new ModelAndView("redirect:/");
3591
3592
         }
3593
3594
3595 9. @ModelAttribute와 data 저장
3596
       1)@ModelAttribute
3597
         -이것은 entity class의 instance를 자동으로 적용할 때 사용
         -인수에는 instance 이름을 지정한다.
3598
3599
         -이것은 전송 form에서 th:object로 지정하는 값이 된다.
         -전송된 form의 값이 자동으로 User instance로 저장된다.
3600
3601
         -따라서 이 annotation을 이용하면 이렇게 쉽게 전송한 data를 저장할 수 있다.
3602
       2)saveAndFlush() method
3603
3604
         -HomeController.java의 아래 code를 보자.
3605
           @RequestMapping(value = "/", method = RequestMethod.POST)
3606
           @Transactional(readOnly=false)
           public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
3607
           mav) {
3608
             repository.saveAndFlush(mydata);
3609
             return new ModelAndView("redirect:/");
3610
           }
3611
3612
       3)미리 설정한 entity는 JpaRepository의 saveAndFlush라는 method를 통해 entity를 영구화한다.
3613
       4)Database를 사용하고 있다면 Database에 그대로 저장된다
3614
3615
3616 10. @Transactional과 transaction
       1)바로 위의 code에서 @Transactional(readOnly=false)가 있다.
3617
3618
       2)이 annotation은 transaction을 위한 것이다.
3619
       3)이 annotation때문에 method내에서 실행되는 database 처리가 일괄적으로 실행되게 된다.
3620
       4)data 변경 처리는 도중에 외부 접속에 의해 data 구조나 내용이 바뀌면 data 일관성에 문제가 발생하게 된다.
3621
       5)이런 문제를 방지하기 위해 transaction이 사용되는 것이다
3622
       6)code를 보면 readOnly=false라고 설정하고 있다.
3623
       7)이 readOnly는 문자 그대로 '읽기 전용(변경 불가)임을 의미한다.
3624
       8)readOnly=false라고 설정하면 변경을 허가하는 transaction이다.
3625
3626
3627 11. Data 초기화 처리
3628
       1)저장한 data는 application을 종료하고 다시 실행하면 지워진다.
3629
       2)HSQLDB는 기본적으로 memory내에 data를 cache하고 있으므로 종료와 함께 지워지는 것이다.
3630
       3)controller에 data를 작성하는 초기화 처리를 넣기로 한다.
       4)HelloController.java code 추가
3631
3632
3633
         @PostConstruct
3634
         public void init(){
3635
           User user1 = new User();
3636
           user1.setUsername("한지민");
3637
           user1.setAge(24);
           user1.setEmail("javaexpert@nate.com");
3638
           user1.setMemo("Hello, Spring JPA");
3639
           repository.saveAndFlush(user1);
3640
3641
           User user2 = new User();
3642
3643
           user2.setUsername("조용필");
```

```
3644
           user2.setAge(66);
3645
           user2.setEmail("aaa@aaa.com");
3646
           user2.setMemo("Good Morning!");
3647
           repository.saveAndFlush(user2);
3648
3649
           User user3 = new User();
3650
           user3.setUsername("이미자");
3651
           user3.setAge(70);
3652
           user3.setEmail("bbb@bbb.com");
3653
           user3.setMemo("Spring Boot is very good.");
3654
           repository.saveAndFlush(user3);
3655
         }
3656
3657
        5)@PostConstruct는 생성자를 통해 instance가 생성된 후에 호출되는 method임을 나타낸다.
3658
        6)Controller는 처음에 한 번만 instance를 만들고 이후에는 해당 instance를 유지한다.
3659
        7)따라서 여기에 test용 data 작성 처리를 해두면 application 실행시에 반드시 한 번 실행되어, data가 준비되는
        것이다.
3660
3661
3662 12. User Find 및 Update 처리하기
        1)src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
3663
3664
        2)File name : edit.html > Finish
3665
3666
          <!DOCTYPE html>
          <html xmlns:th="http://www.thymeleaf.org">
3667
3668
          <head>
          <meta charset="UTF-8">
3669
3670
          <title>edit page</title>
3671
          <style>
3672
         h1 {
3673
           font-size: 18pt;
3674
           font-weight: bold;
3675
           color: gray;
3676
3677
3678
         body {
3679
           font-size: 13pt;
3680
           color: gray;
3681
           margin: 5px 25px;
3682
         }
3683
         tr {
3684
3685
           margin: 5px;
3686
         }
3687
3688
         th {
           padding: 5px;
3689
3690
           color: white;
3691
           background: darkgray;
3692
3693
3694
         td {
3695
           padding: 5px;
3696
           color: black;
           background: #e0e0ff;
3697
3698
          </style>
3699
          </head>
3700
```

```
3701
         <body>
3702
          <h1 th:text="${title}">Edit page</h1>
3703
          3704
            <form method="post" action="/edit" th:object="${formModel}">
              <input type="hidden" name="id" th:value="*{id}" />
3705
3706
              <label for="username">이름</label>
3707
                <input type="text" name="username" th:value="*{username}" />
3708
3709
              3710
              3711
                <label for="age">연령</label>
                <input type="text" name="age" th:value="*{age}" />
3712
3713
              3714
              3715
                <label for="email">메일</label>
                <input type="text" name="email" th:value="*{email}" />
3716
3717
              3718
              3719
                <label for="memo">메모</label>
3720
                <textarea name="memo" th:text="*{memo}" cols="20"
               rows="5"></textarea>
3721
              3722
              3723
                /
3724
3725
              3726
            </form>
3727
          3728
         </body>
3729
         </html>
3730
3731
       3)UserRepository code 추가
3732
3733
        package com.example.biz.dao;
3734
3735
        import java.util.Optional;
3736
3737
        import org.springframework.data.jpa.repository.JpaRepository;
3738
        import org.springframework.stereotype.Repository;
3739
3740
        import com.example.biz.vo.User;
3741
3742
        @Repository("repository")
3743
        public interface UserRepository extends JpaRepository < User, Long > {
3744
          public Optional<User> findById(Long id);
3745
        }
3746
3747
       4)RequestHandler 작성하기
        -HelloController.java code 추가
3748
3749
          @RequestMapping(value = "/edit/{id}", method = RequestMethod.GET)
3750
3751
          public ModelAndView edit(@ModelAttribute User user, @PathVariable int id,
          ModelAndView mav) {
3752
            mav.setViewName("edit");
3753
            mav.addObject("title", "edit mydata.");
3754
            Optional < User > findUser = repository.findById((long) id);
            mav.addObject("formModel", findUser.get());
3755
            return mav;
3756
```

```
3757
           }
3758
3759
           @RequestMapping(value = "/edit", method = RequestMethod.POST)
           @Transactional(readOnly = false)
3760
3761
           public ModelAndView update(@ModelAttribute User user, ModelAndView mav) {
3762
             repository.saveAndFlush(user);
3763
            return new ModelAndView("redirect:/");
3764
3765
       5)접속해서 아래와 같이 URL을 입력하면
3766
3767
3768
         http://localhost:8080/edit/1
3769
3770
         -해당 ID의 data가 표시된다.
3771
         -data를 변경하고 전송해보자.
3772
         -findById는 어디서 구현되는 것일까?
3773
           --repository는 method의 이름을 기준으로 entity 검색 처리를 자동 생성한다.
3774
           --즉 repository에 method 선언만 작성하고, 구체적인 처리를 구현할 필요가 없다.
3775
3776
3777
     13. Entity delete 구현하기
3778
       1)update를 하고 select를 했으니 이번에는 delete를 해 보자.
3779
       2)delete.html template를 작성한다.
3780
       3)src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
3781
       4)File name : delete.html > Finish
3782
3783
         <!DOCTYPE html>
3784
         <html xmlns:th="http://www.thymeleaf.org">
3785
         <head>
         <meta charset="UTF-8">
3786
3787
         <title>delete page</title>
3788
         <style>
3789
         h1 {
3790
           font-size: 18pt;
3791
           font-weight: bold;
3792
           color: gray;
3793
         }
3794
3795
         body {
3796
           font-size: 13pt;
3797
           color: gray;
3798
           margin: 5px 25px;
3799
         }
3800
3801
         td {
           padding: 0px 20px;
3802
           background: #eee;
3803
         }
3804
3805
         </style>
         </head>
3806
3807
         <body>
           <h1 th:text="${title}">Delete page</h1>
3808
3809
           3810
             <form method="post" action="/delete" th:object="${formModel}">
               <input type="hidden" name="id" th:value="*{id}" />
3811
3812
               3813
3814
```

```
3815
3816
              3817
             3818
             3819
3820
             3821
             3822
              3823
             3824
3825
              <input type="submit" value="delete" />
3826
             3827
           </form>
          3828
3829
        </body>
        </html>
3830
3831
3832
      5)RequestHandler 작성
3833
3834
        @RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
3835
          public ModelAndView delete(@PathVariable int id, ModelAndView mav) {
3836
           mav.setViewName("delete");
           mav.addObject("title", "delete mydata.");
3837
3838
           Optional < User > user = repository.findById((long) id);
           mav.addObject("formModel", user.get());
3839
3840
           return mav;
3841
          }
3842
          @RequestMapping(value = "/delete", method = RequestMethod.POST)
3843
3844
          @Transactional(readOnly = false)
          public ModelAndView remove(@RequestParam long id, ModelAndView mav) {
3845
3846
           repository.deleteById(id);
           return new ModelAndView("redirect:/");
3847
          }
3848
3849
3850
      6)접속해서 실행
      http://localhost:8080/delete/2라고 하면 id가 2번이 출력되고 delete button을 누르면 삭제된다.
3851
```