

```
1 HOL : Spring Boot
2 -----
3 Task1. Maven으로 Spring Boot Project 생성하기
4 1. In Package Explorer
5     1)right-click > New > Project > Maven > Maven Project > Next
6     2)[New Maven project] 창에서 > Next
7
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
16        -Artifact Id : springbootdemo
17        -Version : 0.0.1-SNAPSHOT
18        -Package : com.example.springbootdemo
19        -Finish
20
21
22 2. pom.xml 수정
23     <?xml version="1.0" encoding="UTF-8"?>
24
25     <project xmlns="http://maven.apache.org/POM/4.0.0"
26       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
27       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
28         http://maven.apache.org/xsd/maven-4.0.0.xsd">
29
30       <modelVersion>4.0.0</modelVersion>
31
32       <groupId>com.example</groupId>
33       <artifactId>springbootdemo</artifactId>
34       <version>0.0.1-SNAPSHOT</version>
35
36       <name>springbootdemo</name>
37       <!-- FIXME change it to the project's website -->
38       <url>http://www.example.com</url>
39       <parent>
40         <groupId>org.springframework.boot</groupId>
41         <artifactId>spring-boot-starter-parent</artifactId>
42         <version>2.2.6.RELEASE</version>
43       </parent>
44
45       <properties>
46         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
47         <maven.compiler.source>13</maven.compiler.source>
48         <maven.compiler.target>13</maven.compiler.target>
49       </properties>
50
51       <dependencies>
52         <dependency>
53           <groupId>junit</groupId>
54           <artifactId>junit</artifactId>
55           <version>4.13</version>
56           <scope>test</scope>
57         </dependency>
58         <dependency>
59           <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>
65 </dependencies>
66 ...
67 ...
68
69 1)<parent> 설정하기
70 -Spring Boot의 설정 정보를 상속한다.
71 -여기서 지정한 version이 spring boot의 version이 된다.
72 -spring boot의 version을 올리려면 <version> tag 안에 있는 설정 값을 변경한다.
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
84 2)Java Compiler > JDK Compliance > 13 > Apply
85 3)Project Facets > Java 13 > Runtimes tab > Check [jdk-13.0.2]
86 4)Apply and Close
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;
100    import org.springframework.web.bind.annotation.RestController;
101
102    /**
103     * Hello world!
104     *
105     */
106    @RestController
107    @EnableAutoConfiguration
108    public class App {
109        @RequestMapping("/")
110        String home(){
111            return "Hello, World!";
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 application에서 request를 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
125     이 필요없게 된다.
126
127 4)@RequestMapping("/")
128     -이 annotation이 붙으면 이 method가 HTTP 요청을 받아들이는 method임을 나타낸다.
129     -@GetMapping도 가능
130     -@RequestMapping(value="/", method=RequestMethod.GET)과 @GetMapping은 동일하다.
131
132 5)return "Hello World!";
133     -HTTP 응답을 반환한다.
134     -@RestController annotation이 붙은 class에 속한 method에서 문자열을 반환하면 해당 문자열이 그대로
135     HTTP 응답이 되어 출력된다.
136
137 6)SpringApplication.run(App.clas, args);
138     -spring boot applicaton을 실행하는 데 필요한 처리를 main() 안에서 작성한다.
139     -@EnableAutoConfiguration annotation이 붙은 class를 SpringApplication.run()의 첫번째 인자로
140     지정한다.
141
142 6. Web Application 실행하기
143 1)springbootdemo project > right-click > Run As > Spring Boot App
144
145
146
147
148
149
150
151
152
153
154
155
156

```

```

  /\ /  _  ' _ _ _ _ _ ( _ ) _ _ _ _ _ \ \ \ \
(( ) \ _ | ' | ' | | ' V _ | \ \ \ \
 \V _ _ ) | | ) | | | | | ( | | ) ) ) )
  ' | _ _ | . _ | | | | | _ \ , | / / / /
=====|_|=====|___/=/_/_/_/
:: Spring Boot ::      (v2.2.6.RELEASE)

```

```

2020-04-20 21:58:03.208 INFO 14596 --- [          main]
com.example.springbootdemo.App      : Starting App on DESKTOP-1BKHISM with PID
14596 (C:\SpringHome\springbootdemo\target\classes started by devex in
C:\SpringHome\springbootdemo)
2020-04-20 21:58:03.210 INFO 14596 --- [          main]
com.example.springbootdemo.App      : No active profile set, falling back to default
profiles: default
2020-04-20 21:58:03.763 INFO 14596 --- [          main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080
(http)
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 --- [          main]
org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
Tomcat/9.0.33]
2020-04-20 21:58:03.837 INFO 14596 --- [          main]
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
158 2020-04-20 21:58:03.980 INFO 14596 --- [      main]
      o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService
      'applicationTaskExecutor'
159 2020-04-20 21:58:05.842 INFO 14596 --- [      main]
      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.
163 3)SpringApplication.run() method에서 내장 server를 시작했기 때문이다.
164 4)http://localhost:8080/로 접속해보자.
165 5)Web browser에 'Hello, World!'가 출력된다.
166 6)Console에는 아래와 같이 출력된다.
167 2020-04-20 21:59:13.181 INFO 14596 --- [nio-8080-exec-1]
      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'
169 2020-04-20 21:59:13.186 INFO 14596 --- [nio-8080-exec-1]
      o.s.web.servlet.DispatcherServlet : Completed initialization in 5 ms
170
171 7)application을 끝내려면 Ctrl + C를 누르고, '[일괄 작업을 끝내시겠습니까 (Y/N)]?'라는 질문에 'y'를 입력하고
      enter key를 누르면 된다.
172 -또는 빨간색 실행 중지 Button을 click한다.
173 2020-04-20 21:59:47.560 INFO 14596 --- [on(8)-127.0.0.1]
      inMXBeanRegistrar$SpringApplicationAdmin : Application shutdown requested.
174 2020-04-20 21:59:47.562 INFO 14596 --- [on(8)-127.0.0.1]
      o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService
      'applicationTaskExecutor'
175
176 8)여기서 알게 된 사실
177 -설정할 의존 관계의 갯수가 적다.
178 -Java Class 하나만 작성하면 된다.
179 -명령 prompt에서 application을 실행한다.
180
181
182
183 -----
184 Task2. STS로 Spring Boot Application 개발하기
185 1. Package Explorer > right-click > New > Spring Starter Project
186 1)Service URL : http://start.spring.io
187 2)Name : demo
188 3)Type : Maven
189 4)Packaging : jar
190 5)Java Version : 8
191 6)Language : Java
192 7)Group : com.example
193 8)Artifact : demo
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
201 1)Spring Boot Version : 2.2.6
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 ( ) \ _ _ | ' _ | ' _ | ' _ V _ | \ \ \ \
227 \V _ _ | | _ | | | | | | ( _ | ) ) ) )
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)
233 2020-04-20 22:03:17.464 INFO 15496 --- [          main]
com.example.demo.DemoApplication : No active profile set, falling back to default
profiles: default
234 2020-04-20 22:03:18.048 INFO 15496 --- [          main]
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/9.0.33]
237 2020-04-20 22:03:18.120 INFO 15496 --- [          main] o.a.c.c.C.[Tomcat].[localhost].[/]
: Initializing Spring embedded WebApplicationContext
238 2020-04-20 22:03:18.120 INFO 15496 --- [          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 ""
241 2020-04-20 22:03:20.069 INFO 15496 --- [          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;

259 import org.springframework.web.bind.annotation.RequestMapping;

260 import org.springframework.web.bind.annotation.RestController;

261

262 @RestController

263 @EnableAutoConfiguration

264 public class DemoApplication {

265

266 @RequestMapping("/")

267 String home() {

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

288 <https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-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("/")

304 def home() {

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 (() \ _ _ | ' _ | ' _ | ' _ V _ ' _ \ \ \ \

318 \ V _ _) | | _ | | | | | | (_ | |))))

319 ' | _ _ | . _ | _ | | _ | _ \ , | / / / /

320 =====|_|=====|_|_/=/_/_/

321 :: Spring Boot :: (v2.2.4.RELEASE)

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-0]
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-0]
o.apache.catalina.core.StandardService : Starting service [Tomcat]

327 2020-02-18 23:22:22.532 INFO 12048 --- [runner-0]
org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
Tomcat/9.0.30]

328 2020-02-18 23:22:22.539 INFO 12048 --- [runner-0]
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-0]
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]

331 2020-02-18 23:22:22.552 INFO 12048 --- [runner-0]
o.a.catalina.core.AprLifecycleListener : OpenSSL successfully initialized [OpenSSL 1.1.1c
28 May 2019]

332 2020-02-18 23:22:22.667 INFO 12048 --- [runner-0]
org.apache.catalina.loader.WebappLoader : Unknown class loader
[\[org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader\\$DefaultScopeParentClassLoader@6e84d437\]](#) of class [class
org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader\$DefaultScopeParentClassLoader]

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-0]
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 ""
337 2020-02-18 23:22:24.929 INFO 12048 --- [runner-0] o.s.boot.SpringApplication
: 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("/")
353     def home() {
354         def header = "<html><body>"
355         def footer = "</body></html>"
356         def content = "<h1>Hello! Spring Boot with Groovy</h1><p>This is html
content.</p>"
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         <p>This is sample web page.</p>
404     </body>
405
406 </html>

```

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

7)controller 수정하기

```

411 -app.groovy
412     @Grab("thymeleaf-spring5")
413
414     @Controller
415     class App {
416
417         @RequestMapping("/")
418         @ResponseBody
419         def home(ModelAndView mav) {
420             mav.setViewName("home")
421             mav
422         }
423     }

```

8)다시 script 실행

```
$ spring run app.groovy
```

9)<http://localhost:8080>

Hello! Spring Boot with Thymeleaf

This is sample web page.

8. form 전송하기

1)home.html

```

437 <!doctype html>
438 <html lang="en">

```

```

439     <head>
440         <meta charset="UTF-8" />
441         <title>Index Page</title>
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>
446     </head>
447     <body>
448         <h1>Hello!</h1>
449         <p th:text="${msg}">${msg}</p>
450         <form method="post" action="/send">
451             <input type="text" name="text1" th:value="${value}" />
452             <input type="submit" value="Send" />
453         </form>
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     }
469     @RequestMapping(value = "/send", method=RequestMethod.POST)
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)<a href="http://localhost:8080">http://localhost:8080
483 Hello!
484 Please write your name...
485
486 Hello, 한지민!!!
487
488 -----
489 Task4. SPRING INITIALIZR(Maven)
490 1. Visit <a href="http://start.spring.io">http://start.spring.io/
491 2. 설정
492 1)Maven Project
493 2)Java
494 3)2.2.6
495 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
512 2)Click [Browse...] > demo Folder Select > Finish
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) {
567         return "Hello! : " + name;
568     }
569 }
570
571 2)@RestController
572 -이 annotation을 붙이면 web application에서 request를 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
578 -http://localhost:8080/hello?name=한지민
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
598 @ComponentScan(basePackages = {"com.example"})
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     @AllArgsConstructor
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 {
636         @GetMapping("/getUser")
637         public UserVO getUser() {
638             UserVO user = new UserVO();
639             user.setUserid("jimin");
640             user.setName("한지민");
641             user.setGender("여");
642             user.setCity("서울");
643             return user;
644         }
645     }
646

```

647 3)Relaunch demo

```

648     -http://localhost:8080/getUser
649     {"userid":"jimin","name":"한지민","gender":"여","city":"서울"}
650
651

```

652 11. Spring DevTools 사용하기

- 653 1)위처럼 Controller에 새로운 Method가 추가되면 반드시 실행 중인 Application을 중지하고 Application을 재 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.
- 656 4)현재 사용중인 Project에 DevTools를 추가하려면 pom.xml에 추가 Dependency를 추가해야 한다.
- 657 5)pom.xml을 열어서 <dependency> 제일 마지막 Tag 밑에 Ctrl + Space 를 누른다.
- 658 6)Context Menu에서 [Edit Starters...]를 double-click한다.
- 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.
679 3)하지만 code가 지저분해지고 모든 VO class와 JPA에서 사용할 Domain Class에 이런 Method를 반복적으로
    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.
686 10)[Edit Spring Boot Starters]창에서 Developer Tools > Lombok 체크한다.
687 11)그리고 [OK] 클릭한다.
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.java로 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 {
719         private String userid;
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     -@RequiredArgsConstructor
732         --모든 Member 변수를 초기화하는 생성자를 생성
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, @EqualsAndHashCode 모두
           생성.
739
740
741
742 -----
743 Task5. SPRING INITIALIZER(Gradle)
744 1. Visit http://start.spring.io/
745 2. 설정
746     1)Gradle Project
747     2)Java
748     3)2.2.4
749     4)Group : com.example
750     5)Artifact : demoweb
751     6)Name : demoweb
752     7)Description : Demo project for Spring Boot
753     8)Package Name : com.example.demoweb
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
764     1)In STS, Package Explorer > right-click > Import > Gradle > Existing Gradle Project >
       Next > Next
765     2)Click [Browse...] > demoweb Folder > Select Folder > Next > Finish
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     }

```

```
773
774     group = 'com.example'
775     version = '0.0.1-SNAPSHOT'
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.
809     Thu Nov 07 16:06:13 KST 2019
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
820     import org.springframework.web.bind.annotation.GetMapping;
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     }
```



```
830     }
831
832     3)Relaunch demo
833     4)http://localhost:8080/
834     Hello, Spring Boot World
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
845         -Artifact Id : mybootstarter
846         -Version : 0.0.1-SNAPSHOT
847         -Package : com.example.mybootstarter
848         -Finish
849
850
851 2. Project Facets 수정하기
852     1)mybootstarter Project > right-click > Properties > Project Facets
853     2)Java 1.8 > Runtimes Tab > Apache Tomcat v9.0, jdk1.8.0_241 check
854     3)Apply and Close click
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     <!--
862     https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-autoconfigure
863     -->
864     <dependency>
865         <groupId>org.springframework.boot</groupId>
866         <artifactId>spring-boot-autoconfigure</artifactId>
867         <version>2.2.4.RELEASE</version>
868     </dependency>
869
870     3)pom.xml > right-click > Run As > Maven install
871     [INFO] BUILD SUCCESS
872
873 4. dependencyManagement 추가
874     1)앞으로 추가되는 Library들의 Version을 일괄적으로 관리하기 위해 pom.xml에 Code 추가
875     2)Mvnrepository에서 'spring boot dependencies'로 검색
876     3)Spring Boot Dependencies에서 3.0.1.RELEASE
877     4)아래의 Code를 pom.xml의 제일 아래에 추가
878
879     <!--
880     https://mvnrepository.com/artifact/org.apache.camel.springboot/camel-spring-boot-depe
881     ndencies -->
882     <dependency>
883         <groupId>org.apache.camel.springboot</groupId>
884         <artifactId>camel-spring-boot-dependencies</artifactId>
885         <version>3.0.1</version>
886         <scope>provided</scope>
```

```

884     </dependency>
885
886 5)pom.xml
887
888     <dependencyManagement>
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>
897             </dependency>
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>
906         <version>1.18.12</version>
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.
916 2)com.example.util package 생성
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
929     public class JdbcConnectionManager {
930         private String driverClass;
931         private String url;
932         private String username;
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();
962             manager.setDriverClass("oracle.jdbc.driver.OracleDriver");
963             manager.setUrl("jdbc:oracle:thin:@localhost:1521:XE");
964             manager.setUsername("hr");
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.UserAutoConfiguration
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
992     packaging한 jar 파일이 등록되어 있을 것이다.
993
994 10. Starter와 자동 설정 사용하기
995     1)사용자 정의 Starter가 Maven Repository에 등록됐으면 이제 이 Starter를 이용하여 Application을 만들수
```

있다.

2)위에서 생성한 Project의 pom.xml에서 아래의 Code를 복사한다.

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

3)위의 Task4 에서 생성한 demo Project의 pom.xml의 <dependency>에 붙여넣는다.

```
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
</dependency>
```

<!-- 아래 코드 추가 -->

```
<dependency>
<groupId>com.example</groupId>
<artifactId>mybootstarter</artifactId>
<version>0.0.1-SNAPSHOT</version>
</dependency>
```

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

[INFO] BUILD SUCCESS

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.

7)demo Project에 com.example.service package를 생성한다.

8)com.example.service.JdbcConnectionManagerRunner Class를 생성한다.

```
package com.example.service;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.boot.ApplicationArguments;
```

```
import org.springframework.boot.ApplicationRunner;
```

```
import org.springframework.stereotype.Service;
```

```
import com.example.util.JdbcConnectionManager;
```

```
@Service
```

```
public class JdbcConnectionManagerRunner implements ApplicationRunner {
```

```
    @Autowired
```

```
    private JdbcConnectionManager connectionManager;
```

```
    @Override
```

```
    public void run(ApplicationArguments args) throws Exception {
```

```
        System.out.println("Connection Manager : " + this.connectionManager.toString());
```

```
    }
```

```
}
```

9)위에서 생선한 JdbcConnectionManagerRunner는 Container가 Component Scan하도록 @Service 를 추가했다.

10)ApplicationRunner Interface를 구현했기 때문에 JdbcConnectionManagerRunner 객체가 생성되자 마자 Container에 의해서 run() 가 자동으로 실행된다.

11)demo Project > right-click > Run As > Spring Boot App

12)Console에 다음과 같은 출력이 나오면 자동설정이 정상적으로 동작했다는 의미이다.

```
Connection Manager : JdbcConnectionManager
[driverClass=oracle.jdbc.driver.OracleDriver,
```

1049 `url=jdbc:oracle:thin:@localhost:1521:XE, 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;`

1064 `import org.springframework.context.annotation.Configuration;`

1065 `import com.example.util.JdbcConnectionManager;`

1066

1067 `@Configuration`

1068 `public class UserConfiguration {`

1069

1070 `@Bean`

1071 `public JdbcConnectionManager getJdbcConnectionManager() {`

1072 `JdbcConnectionManager manager = new JdbcConnectionManager();`

1073 `manager.setDriverClass("org.mariadb.jdbc.Driver");`

1074 `manager.setUrl("jdbc:mariadb://localhost:3306/test");`

1075 `manager.setUsername("root");`

1076 `manager.setPassword("javamariadb");`

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.

1083 The bean 'getJdbcConnectionManager', defined in class path resource

1084 [com/example/config/UserAutoConfiguration.class], could not be registered. A bean with that

1085 name has already been defined in class path resource

1086 [com/example/config/UserConfiguration.class] and overriding is disabled.

1087 -즉, Memory에 같은 Type의 bean이 두 개가 등록되어 충돌이 발행했다는 메시지이다.

1088 -이 문제를 해결하기 위해서는 새로 생성된 bean이 기존에 등록된 bean을 덮어쓸 수 있도록 해야 한다.

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

1094 1. application.properties 파일에 작성한 한글이 정상적으로 보이지 않으면 [Properties Editor] plugin을 설

1095 치하면 된다.

1096 2. Help > Install New Software... > Add...

1097 3. Name : Properties Editor

1098 4. Location : <http://propedit.sourceforge.jp/eclipse/updates>

1099 5. Add

1100 6. 이렇게 설치하는 이유는 현재 Eclipse Marketplace에서 'Properties Editor'로 검색되지 않기 때문이다.

1101 7. 목록에서 [PropertiesEditor]만 Check하고 Next

1102 8. 다른 Plugin 설치와 마찬가지로 계속 설치를 진행한다.

1103 9. 설치과정이 마치면 STS를 재 시작하고 application.properties file을 선택하고 Mouse right-click >

Open With > PropertiesEditor

10. 한글이 깨지지 않고 정상적으로 보이는 것을 볼 수 있다.

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

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

-이 문제를 해결하기 위한 Annotation이 바로 @Conditional이다.

-@Conditional Annotation은 조건에 따라 새로운 객체를 생성할지 안할지를 결정할 수 있다.

2)@Conditional Annotation 사용하기

-@SpringBootApplication은 @EnableAutoConfiguration과 @ComponentScan을 포함하고 있다.

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

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

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

-mybootstarter Project의 com.example.config.UserAutoConfiguration Class에 @ConditionalOnMissingBean Annotation을 적용한다.

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

```
import org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import com.example.util.JdbcConnectionManager;
```

```
@Configuration
```

```
public class UserAutoConfiguration {
```

```
    @Bean
```

```
    @ConditionalOnMissingBean
```

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

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

```
        manager.setDriverClass("oracle.jdbc.driver.OracleDriver");
```

```
        manager.setUrl("jdbc:oracle:thin:@localhost:1521:XE");
```

```
        manager.setUsername("hr");
```

```
        manager.setPassword("hr");
```

```
        return manager;
```

```
    }
```

```
}
```

-@ConditionalONMissingBean은 등록하려는 Bean이 Memory에 없는 경우에만 현재의 Bean 등록을 처리하도록 한다.

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

-이제 모두 저장하고 mybootstarter Project를 다시 install한다.

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

```
[INFO] BUILD SUCCESS
```

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

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

3)Property File 이용하기

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

-Lab을 위해서 demo Project의 com.example.config.UserConfiguration.java의 @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");
1160             manager.setUrl("jdbc:mariadb://localhost:3306/test");
1161             manager.setUsername("root");
1162             manager.setPassword("javamariadb");
1163             return manager;
1164         }
1165     }
1166
1167     -demo Project의 application.properties 파일에 다음 코드를 추가한다.
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;
1182     import java.sql.DriverManager;
1183     import org.springframework.boot.context.properties.ConfigurationProperties;
1184
1185     @ConfigurationProperties(prefix="user.jdbc")
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         }
1204         public String getUsername() {
1205             return username;
1206         }

```

```

1207     public void setUsername(String username) {
1208         this.username = username;
1209     }
1210     public String getPassword() {
1211         return password;
1212     }
1213     public void setPassword(String password) {
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-processor to pom.xml
link를 click한다.
1220 -이렇게 하면 자동으로 pom.xml에 다음과 같은 dependency가 추가된다.
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
1231 -이제 mybootstarter Project의 com.example.config.UserAutoConfiguration Class를 수정한다.
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() {
1253             JdbcConnectionManager manager = new JdbcConnectionManager();
1254             manager.setDriverClass(this.properties.getDriverClass());
1255             manager.setUrl(this.properties.getUrl());
1256             manager.setUsername(this.properties.getUsername());
1257             manager.setPassword(this.properties.getPassword());
1258             return manager;
1259         }
1260     }
1261
1262 -이제 mybootstarter Project를 다시 Install 한다.
1263     --mybootstarter Project > right-click > Run As > Maven install

```



```

1264      [INFO] BUILD SUCCESS
1265      -demo Project를 Refresh하고 다시 실행한다.
1266      --다시 Oracle 설정 정보가 나오는 것을 알 수 있다.
1267      Connection Manager : JdbcConnectionManager
1268      [driverClass=oracle.jdbc.driver.OracleDriver,
1269      url=jdbc:oracle:thin:@localhost:1521:XE, username=hr, password=hr]
1269      -만일 Database가 다시 MaraiDB로 변경된다면 demo Project의 application.properties를 다음과 같이
      변경하면 된다.
1270      ## Bean Overriding 설정
1271      spring.main.allow-bean-definition-overriding=true
1272      ## 데이터 소스 : Oracle
1273      #user.jdbc.driverClass=oracle.jdbc.driver.OracleDriver
1274      #user.jdbc.url=jdbc:oracle:thin:@localhost:1521:XE
1275      #user.jdbc.username=hr
1276      #user.jdbc.password=hr
1277      ## 데이터 소스 : MariaDB
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.
1295      4)Main page에서 Download의 All Platforms (zip, 8MB) Link를 Click하여 압축파일을 Download한다.
1296      5)h2-2019-10-14.zip 압축을 풀고 h2 Folder를 C:/Program Files로 이동한다.
1297      6)h2/bin의 h2w.bat를 실행하면 Browser기반의 관리 Console이 열린다.
1298
1299      7)Tray에 있는 H2 Database Engine > right-click > Create a Database...을 실행하여 다음의 각 항목에
      값을 입력하고 [Create] button을 click한다.
1300      -Database path : ./test
1301      -Username : sa
1302      -Password : javah2
1303      -Password confirmation : javah2
1304      -Create button click
1305      -----
1306      Database was created successfully.
1307      JDBC URL for H2 Console:
1308      jdbc:h2:./test
1309
1310      8)각 항목의 정보를 입력하고 [Test Connection] 클릭해본다.
1311      --Driver Class : org.h2.Driver
1312      --JDBC URL : jdbc:h2:~/test
1313      --User Name : sa
1314      --Password : javah2
1315
1316      9)연결이 성공하면 Web Console이 열린다.
1317
1318

```

1319 2. JPA Project Installation
1320 1)In STS, Help > Install New Software
1321 2)Work with : <https://download.eclipse.org/releases/2019-12>
1322 3)Filter : jpa
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
1334 1)Package Explorer > right-click > Maven Project
1335 -Next
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 : jpademo
1342 -Version : 0.0.1-SNAPSHOT
1343 -Package : com.example.jpademo
1344 -Finish
1345
1346 3)Project Facets 변환
1347 -jpademo Project > right-click > Properties > Project Facets
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
1354 ---JPA implementation
1355 Type : Disable Library Configuration
1356 --OK
1357 -Apply and Close
1358
1359 4)Maven Project를 JPA Project로 변경하면 src/main/java하위에 META-INF/persistence.xml JPA 환경설정 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>
1381         <groupId>com.h2database</groupId>
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
1387     -'lombok'으로 검색하여
1388     <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
1389     <dependency>
1390         <groupId>org.projectlombok</groupId>
1391         <artifactId>lombok</artifactId>
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
1406     선언하면 된다.
1407     6)다만, Eclipse의 JPA Perspective가 제공하는 Entity 생성 기능을 사용하면 Entity를 생성함과 동시에 영속성
1408     설정 파일(persistence.xml)에 자동으로 Entity가 등록된다.
1409     7)src/main/java Folder에 com.example.jpdemo package > right-click > New > JPA Entity
1410     8)다음 항목의 값을 입력 후 Finish 클릭
1411     -Java package : com.example.domain
1412     -Class name : User
1413
1414     9)META-INF/persistence.xml 파일이 자동으로 수정되었다.
1415     <?xml version="1.0" encoding="UTF-8"?>
1416     <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
1417     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1418     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
1419     http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">
1420     <persistence-unit name="jdemo">
1421         <class>com.example.domain.User</class>
1422     </persistence-unit>
1423 </persistence>
1424
1425     10)이제 방금 생성한 User Class를 수정한다.
1426
1427     package com.example.domain;
1428
1429     import java.io.Serializable;
1430
1431     import javax.persistence.Entity;
1432     import javax.persistence.Id;
1433     import javax.persistence.Table;

```

```

1432
1433 import lombok.Getter;
1434 import lombok.Setter;
1435 import lombok.ToString;
1436
1437 /**
1438  * Entity implementation class for Entity: User
1439  *
1440  */
1441 @Entity
1442 @Table
1443 @Getter
1444 @Setter
1445 @ToString
1446 public class User implements Serializable {
1447
1448     @Id
1449     private String userid;
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

```

11)다음은 JPA 사용하는 주요 Annotation을 설명한 것이다.

```

1463 -@Entity
1464     --Entity Class 임을 설명
1465     --기본적으로 Class의 이름과 동일한 Table과 Mapping된다.
1466 -@Table
1467     --Entity의 이름과 Table의 이름이 다를 경우, name 속성을 이용하여 Mapping 한다.
1468     --이름이 동일하면 생략 가능
1469 -@Id
1470     --Table의 primary key와 Mapping한다.
1471     --Entity의 필수 Annotation으로서 @Id가 없으면 Entity는 사용 불가
1472 -@GeneratedValue
1473     --@Id가 선언된 Field에 기본 키 값을 자동으로 할당
1474
1475
1476

```

1477 6. JPA main 설정 파일 작성

```

1478 1)META-INF/persistence(JPA의 main 환경설정 파일) 수정
1479 <?xml version="1.0" encoding="UTF-8"?>
1480 <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
1481     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1482     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
1483     http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">
1484     <persistence-unit name="jpademo">
1485         <class>com.example.domain.User</class>
1486         <properties>
1487             <!-- 필수 속성 -->
1488             <property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>
1489             <property name="javax.persistence.jdbc.user" value="sa"/>

```

```

1490     <property name="javax.persistence.jdbc.password" value="javah2"/>
1491     <property name="javax.persistence.jdbc.url" value="jdbc:h2:~/test"/>
1492     <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
1493     <!-- Option 속성 -->
1494     <property name="hibernate.show_sql" value="true"/>
1495     <property name="hibernate.format_sql" value="true"/>
1496     <property name="hibernate.use_sql_comments" value="false"/>
1497     <property name="hibernate.id.new_generator_mappings" value="true"/>
1498     <property 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.
1505 4)이 속성을 H2Dialect Class로 설정하면 H2용 SQL이 생성되고, OracleDialect로 변경하면 Oracle용 SQL이
    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");
1526                 user.setUsername("한지민");
1527                 user.setGender("여");
1528                 user.setAge(24);
1529                 user.setCity("서울");
1530                 // 등록
1531                 em.persist(user);
1532             } catch (Exception e) {
1533                 e.printStackTrace();
1534             } finally {
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
1549     ERROR: Database may be already in use: null. Possible solutions: close all other
1550         connection(s); use the server mode [90020-200]
1551 -작업관리자에서 javaw.exe 작업끝내기를 수행한다.
1552
1553 3)실행 결과
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
1577 1)JPA가 실제 Table에 등록/수정/삭제 작업을 처리하기 위해서는 해당 작업이 반드시 Transaction안에서 수행되어
1578     야 한다.
1579 2)만약 Transaction을 시작하지 않았거나 등록/수정/삭제 작업 이후에 Transaction을 종료하지 않으면 요청한 작
1580     업이 실제 Database에 반영되지 않는다.
1581 3)JPAClient.java를 수정한다.
1582
1583     import javax.persistence.EntityManager;
1584     import javax.persistence.EntityManagerFactory;
1585     import javax.persistence.EntityTransaction;
1586     import javax.persistence.Persistence;
1587
1588     import com.example.domain.User;
1589
1590     public class JPAClient {
1591         public static void main(String[] args) {
1592             // EntityManager 생성
1593             EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
1594             EntityManager em = emf.createEntityManager();
1595             //Transaction 생성
1596             EntityTransaction tx = em.getTransaction();
1597             try {
1598                 //Transaction 시작
1599                 tx.begin();
1600
1601                 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) {
1611         e.printStackTrace();
1612         // Transaction rollback
1613         tx.rollback();
1614     } finally {
1615         em.close();
1616         emf.close();
1617     }
1618 }
1619 }
```

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     (?, ?, ?, ?, ?)
```

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

1641

1642 4)다음으로 변경한다.

1643

```
1643 <property name="hibernate.hbm2ddl.auto" value="update"/>
```

1644

1645 5)이렇게 변경하면 새롭게 생성하지 않고 기존의 Table을 재사용한다.

1646

1647 6)다음의 Data를 수행해서 3명의 User를 Insert한다.

1648

1648 chulsu, 34, 부산, 남, 김철수

1649

1649 younghee, 44, 대전, 여, 이영희

1650

1651

1652 10. Data 검색

1653

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 {
1663     public static void main(String[] args) {
1664         // EntityManager 생성
1665         EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
1666         EntityManager em = emf.createEntityManager();
1667         try {
1668             // User 검색
1669             User user = em.find(User.class, "jimin");
1670             System.out.println("jimin --> " + user);
1671         } catch (Exception e) {
1672             e.printStackTrace();
1673         } finally {
1674             em.close();
1675             emf.close();
1676         }
1677     }
1678 }

```

2) 실행

```

1682 Hibernate:
1683 select
1684     user0_.userid as userid1_0_0_,
1685     user0_.age as age2_0_0_,
1686     user0_.city as city3_0_0_,
1687     user0_.gender as gender4_0_0_,
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]

```

11. Entity 수정

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 {
1714             // Transaction 시작
1715             tx.begin();

```



```

1716         // 수정할 User 조회
1717         User user = em.find(User.class, "younghee");
1718         user.setCity("광주");
1719         user.setAge(55);
1720         // Transaction commit
1721         tx.commit();
1722     } catch (Exception e) {
1723         e.printStackTrace();
1724         // Transaction rollback
1725         tx.rollback();
1726     } finally {
1727         em.close();
1728         emf.close();
1729     }
1730 }
1731 }

```

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     from
1742         User user0_
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=?

```

12. Entity 삭제

1)JPAClient.java 수정

```

1759
1760 import javax.persistence.EntityManager;
1761 import javax.persistence.EntityManagerFactory;
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("jpademo");
1771         EntityManager em = emf.createEntityManager();
1772         // Transaction 생성
1773         EntityTransaction tx = em.getTransaction();

```

```
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
1782         // Transaction commit
1783         tx.commit();
1784     } catch (Exception e) {
1785         e.printStackTrace();
1786         // Transaction rollback
1787         tx.rollback();
1788     } finally {
1789         em.close();
1790         emf.close();
1791     }
1792 }
1793 }
```

2)실행

```
1796 Hibernate:
1797     select
1798         user0_.userid as userid1_0_0_,
1799         user0_.age as age2_0_0_,
1800         user0_.city as city3_0_0_,
1801         user0_.gender as gender4_0_0_,
1802         user0_.username as username5_0_0_
1803     from
1804         User user0_
1805     where
1806         user0_.userid=?
1807 Hibernate:
1808     delete
1809     from
1810         User
1811     where
```

1812

1813

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 {
1838     // Transaction 시작
1839     tx.begin();
1840
1841     User user = new User();
1842     user.setUserid("hojune");
1843     user.setUsername("이호준");
1844     user.setAge(30);
1845     user.setGender("남");
1846     user.setCity("수원");
1847
1848     // User 등록
1849     em.persist(user);
1850
1851     // Transaction commit
1852     tx.commit();
1853
1854     // 여러 Record 조회
1855     String jpql = "SELECT u FROM User u ORDER BY u.userid DESC";
1856     List<User> userList = em.createQuery(jpql, User.class).getResultList();
1857     for (User usr : userList) {
1858         System.out.println(usr);
1859     }
1860 } catch (Exception e) {
1861     e.printStackTrace();
1862     // Transaction rollback
1863     tx.rollback();
1864 } finally {
1865     em.close();
1866     emf.close();
1867 }
1868 }
1869 }

```

4)실행

```

1872 Hibernate:
1873 insert
1874 into
1875     User
1876     (age, city, gender, username, userid)
1877 values
1878     (?, ?, ?, ?, ?)
1879 Hibernate:
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_,
1885     user0_.username as username5_0_
1886 from
1887     User user0_
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]
1893     User [userid=chulsu, username=김철수, gender=남, age=34, city=부산]
```

```
1894
1895
1896
1897 -----
```

1898 Task8. 정적 Page 만들기

1899 1. Spring Boot project 생성

1900 1)Package Explorer > right-click > New > Spring Starter Project

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

1906 -Java Version : 8

1907 -Language : Java

1908 -Group : com.example

1909 -Artifact : springweb

1910 -Version : 0.0.1-SNAPSHOT

1911 -Description : Demo project for Spring Boot

1912 -Package : com.example.biz

1913 -Next

1914

1915 3)다음의 각 항목을 선택한 후 Finish 클릭

1916 -Spring Boot Version : 2.2.6

1917 -Developer Tools > Spring Boot DevTools

1918 -Web > Spring Web

1919

1920

1921 2. Controller 생성

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 {

1933 @GetMapping("/")

1934 public ModelAndView home(ModelAndView mav) {

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
1953     <!DOCTYPE html>
1954     <html>
1955     <head>
1956     <meta charset="UTF-8">
1957     <title>Home page</title>
1958     <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css" />
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             
1969         </div>
1970         <div class="jumbotron">
1971             <h1>Hello, Spring Boot World</h1>
1972             <p>...</p>
1973             <p>
1974                 <a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>
1975             </p>
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. http://localhost:8080/
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.
1991 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 : springjspdemo
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
2010 3)각 항목 선택 후 Finish 클릭
2011     -Spring Boot Version : 2.2.6
2012     -Web > Spring Web > Finish
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</groupId>
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
2037 3)WEB-INF folder 안에 jsp folder 생성
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
2047 10. jsp folder 안에 jsp file 생성
2048 1)index.jsp
2049
2050     <%@ page language="java" contentType="text/html; charset=UTF-8"
2051     pageEncoding="UTF-8"%>
2052     <%@ page import="java.util.Date, java.text.SimpleDateFormat" %>
2053     <!DOCTYPE html>
2054     <html>
2055     <head>
2056         <meta charset="UTF-8">
2057         <title>Insert title here</title>
2058     </head>
2059     <body>
```

```
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

2067 1)com.example.biz > right-click > New > Click

2068 2)Name : HomeController

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

2087 <http://localhost:8080/>

2088

2089 Index page

2090 2020년 04월 20일

2091

2092

2093

2094 -----

2095 Task10. thymeleaf template 사용하기

2096 1. Spring Boot project 생성

2097 1)Package Explorer > right-click > New > Spring Starter Project

2098 2)다음의 각 항목 입력 후 Next 클릭

2099 -Service URL :<http://start.spring.io>

2100 -Name : MyBootWeb

2101 -Type : Maven

2102 -Packaging : jar

2103 -Java Version : 8

2104 -Language : Java

2105 -Group : com.example

2106 -Artifact : MyBootWeb

2107 -Version : 0.0.1-SNAPSHOT

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

2133 -<http://localhost:8080/>

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

2140 There was an unexpected error (type=Not Found, status=404).

2141 No message available

2142

2143

2144 4. Controller 작성하기

2145 1)com.example.biz > right-click > New > Class

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

2157 @RequestMapping("/")

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 <= num ; i++) result += i;
2178     return "total : " + result;
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

2211 @AllArgsConstructor

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 {

2223 String [] names = {"조용필", "이미자", "설운도"};

2224 int [] ages = {56, 60, 70};

2225 String [] addresses = {"서울특별시", "부산광역시", "대전광역시"};

2226

2227 @RequestMapping("/{userid}")

2228 public Student index(@PathVariable int userid) {

2229 return new Student(userid, names[userid], ages[userid], addresses[userid]);

2230 }

2231 }

2232

2233 3)<http://localhost:8080/1>

```
2234     {"userid":1,"name":"이미자","age":60,"address":"부산광역시"}
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
2259     public class HelloController {
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>
2274             <meta charset="UTF-8" /> <!--반드시 종결 tag 필요 -->
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             <p>This is sample web page.</p>
2284         </body>
2285     </html>
2286
2287 4)&a href="http://localhost:8080/">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   <p class="msg" th:text="${msg}"></p>
2298 </body>
2299
2300 -HelloController.java 수정
2301 @Controller
2302 public class HelloController {
2303
2304   @RequestMapping("/{num}")
2305   public String index(@PathVariable int num, Model model) {
2306     int result = 0;
2307     for(int i = 1 ; i <= num ; i++) result += i;
2308     model.addAttribute("msg", "total : " + result);
2309     return "index";
2310   }
2311 }
2312
2313 6)<a href="http://localhost:8080/100">http://localhost:8080/100
2314
2315 Hello! Spring Boot with Thymeleaf
2316 total : 5050
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 <= num ; i++) result += i;
2327     mav.addObject("msg", "total : " + result);
2328     mav.setViewName("index");
2329     return mav;
2330   }
2331 }
2332
2333 8)<a href="http://localhost:8080/100">http://localhost:8080/100
2334
2335 Hello! Spring Boot with Thymeleaf
2336 total : 5050
2337
2338 9)form 사용하기
2339 -index.html 수정
2340
2341 <!doctype html>
2342 <html lang="en">
2343   <head>
2344     <meta charset="UTF-8" />
2345     <title>Index Page</title>
2346     <style type="text/css">
2347       h1 { font-size:18pt; font-weight:bold; color:gray; }
2348       body { font-size:13pt; color:gray; margin:5px 25px; }
2349     </style>
```

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

```

2407     <pre th:text="${msg}">Please wait...</pre>
2408     <form method="post" action="/">
2409     <div>
2410         <input type="checkbox" id="ckeck1" name="check1" /> <label for="ckeck1">체크
2411         </label>
2412     </div>
2413     <div>
2414         <input type="radio" id="male" name="gender" value="male" /> <label for="male">
2415         남성</label>
2416     </div>
2417     <div>
2418         <input type="radio" id="female" name="gender" value="female" /> <label
2419         for="female">여성</label>
2420     </div>
2421     <div>
2422         <select name="selOs" size="4">
2423             <option>--선택--</option>
2424             <option value="Windows">windows</option>
2425             <option value="MacOS">MacOS</option>
2426             <option value="Linux">Linux</option>
2427         </select>
2428     </div>
2429     <select name="selEditors" size="4" multiple="multiple">
2430         <option>--선택--</option>
2431         <option value="Notepad">Notepad</option>
2432         <option value="Editplus">Editplus</option>
2433         <option value="Visual Studio Code">Visual Studio Code</option>
2434         <option value="Sublime Text">Sublime Text</option>
2435         <option value="Eclipse">Eclipse</option>
2436     </select>
2437     </div>
2438     <input type="submit" value="전송" />
2439     </form>
2440 </body>
2441 </html>

```

12)HelloController.java 수정

```

2441 @Controller
2442 public class HelloController {
2443
2444     @RequestMapping(value = "/", method = RequestMethod.GET)
2445     public ModelAndView index(ModelAndView mav) {
2446
2447         mav.setViewName("index");
2448         mav.addObject("msg", "값을 입력후 전송버튼을 눌러주세요.");
2449         return mav;
2450     }
2451
2452     @RequestMapping(value = "/", method = RequestMethod.POST)
2453     public ModelAndView send(@RequestParam(value="check1", required=false) boolean
2454     check1,
2455         @RequestParam(value="gender", required=false) String gender,
2456         @RequestParam(value="selOs", required=false) String selOs,
2457         @RequestParam(value="selEditors", required=false) String [] selEditors,
2458         ModelAndView mav) {
2459         String result = "";
2460         try {
2461             result = "check : " + check1 + ", gender : " + gender + ", OS : " + selOs +

```

```

2460         "\nEditors : ";
2461     }catch(NullPointerException ex) {}
2462     try {
2463         result += selEditors[0];
2464         for(int i = 1 ; i < selEditors.length ; i++) result += ", " + selEditors[i];
2465     }catch(NullPointerException ex) {
2466         result += "null";
2467     }
2468
2469     mav.addObject("msg", result);
2470     mav.setViewName("index");
2471     return mav;
2472 }
2473 }
2474

```

2475 13)<http://localhost:8080>

2478 12. Redirect

2479 1)index.html 수정

```

2481 <body>
2482 <h1>Hello! index.</h1>
2483 </body>
2484

```

2485 2>HelloController.java 수정

```

2486 @Controller
2487 public class HelloController {
2488
2489     @RequestMapping("/")
2490     public ModelAndView index(ModelAndView mav) {
2491         mav.setViewName("index");
2492         return mav;
2493     }
2494
2495     @RequestMapping("/other")
2496     public String other() {
2497         return "redirect:/";
2498     }
2499
2500     @RequestMapping("/home")
2501     public String home() {
2502         return "forward:/";
2503     }
2504 }
2505

```

2507 3)redirect와 forward 차이점 구분하기

2512 Task11. thymeleaf template 사용하기2

2513 1. Spring Boot project 생성

2514 1)Package Explorer > right-click > New > Spring Starter Project

2516 2)다음 각 항목의 값을 입력 후 Next 클릭

```
2517 -Service URL :http://start.spring.io
2518 -Name : templatedemo
2519 -Type : Maven
2520 -Packaging : Jar
2521 -Java Version : 8
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
2531 -Select
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
2551     @GetMapping("/")
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
2561 <!DOCTYPE html>
2562 <html>
2563     <head>
2564         <meta charset="UTF-8" />
2565         <title>Insert title here</title>
2566     </head>
2567     <body>
2568         <h1>Hello Page</h1>
2569         <p th:text="${new java.util.Date().toString()}"></p>
2570     </body>
2571 </html>
2572
2573 4)실행
2574 --http://localhost:8080
```

```
2575
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         <p th:text="${#dates.format(new java.util.Date(), 'dd/MM/yyyy HH:mm')}"></p>
2586         <p th:text="${#numbers.formatInteger(1234,7)}"></p>
2587         <p th:text="${#strings.toUpperCase('Welcome to Spring.')}"></p>
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>
2604         <h1>Hello page</h1>
2605         <p th:text="from parameter... id=' + ${param.id[0]} + ',name=' +
2606             param.name[0]}"></p>
2607     </body>
2608
2609     2)HomeController.java를 수정한다.
2610
2611     @RequestMapping("/")
2612     public ModelAndView index(ModelAndView mav) {
2613         mav.setViewName("index");
2614         return mav;
2615     }
2616
2617     @RequestMapping("/home")
2618     public String home() {
2619         return "forward:/";
2620     }
2621
2622     3)그리고 id와 name을 query string으로 지정해서 접속한다.
2623     -http://localhost:8080/home/?id=javaexpert&name=Springboot
2624     -결과는 아래와 같다.
2625     Helo page
2626     from parameter... id=javaexpert,name=Springboot
2627
2628     4)controller를 거치지 않고 template내에서 직접 전달된 값을 사용할 수 있다.
2629     5)중요한 것은 param내의 id나 name 배열에서 첫 번째 요소를 지정해서 추출한다는 것이다.
2630     6)그 이유는 query string으로 값을 전송할 때 같은 이름의 값을 여러 개 전송하기 위해서다.
2631     7)예를 들면, http://localhost:8080/home/?id=123&id=456&name=javaexpert&name=peter
2632     8)이렇게 하면 param에서 추출하는 id와 name이 각각 {123,456}, {javaexpert,peter}가 되는 것이기 때문
```


이다.

2632 9)여기서 사용하고 있는 `th:text`의 값을 보면 큰따옴표 안에 다시 작은 따옴표를 사용해서 값을 작성하고 있다.

2633 10)이것은 OGNL로 `text literal`을 작성할 때 사용하는 방식이다.

2634 11)이렇게 하면 다수의 `literal`을 연결할 때 큰 따옴표안에 작은 따옴표 `literal`을 사용할 수 있게 된다.

2635

2636 `th:text="one two three"`

2637 `th:text="'one' + ' two ' + 'three'"`

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 `<p th:text="#{content.message}"></p>`

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 `<p><a th:href="@{/home/{orderId}(orderId=${param.id[0]})}">link</p>`

2665 `</body>`

2666

2667 2)접속할 때 query string에 id를 지정한다.

2668 3)예를 들어 <http://localhost:8080/?id=123>에 접속하면 link에는 /home/123이 설정된다.

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;`

2684 `private String username;`

2685 `private int age;`

2686 `}`

2687

2688 3)HomeController.java 수정

```

2689
2690 @RequestMapping("/")
2691 public ModelAndView index(ModelAndView mav) {
2692     mav.setViewName("index");
2693     mav.addObject("msg","Current data.");
2694     Member obj = new Member(123, "javaexpert",24);
2695     mav.addObject("object",obj);
2696     return mav;
2697 }
2698
2699 4)index.html 수정
2700
2701 <!DOCTYPE HTML>
2702 <html xmlns:th="http://www.thymeleaf.org">
2703     <head>
2704         <title>top page</title>
2705         <meta charset="UTF-8" />
2706         <style>
2707             h1 { font-size:18pt; font-weight:bold; color:gray; }
2708             body { font-size:13pt; color:gray; margin:5px 25px; }
2709             tr { margin:5px; }
2710             th { padding:5px; color:white; background:darkgray; }
2711             td { padding:5px; color:black; background:#e0e0ff; }
2712         </style>
2713     </head>
2714     <body>
2715         <h1 th:text="#{content.title}">Hello page</h1>
2716         <p th:text="${msg}">message.</p>
2717         <table th:object="${object}">
2718             <tr><th>ID</th><td th:text="*{id}"></td></tr>
2719             <tr><th>NAME</th><td th:text="*{username}"></td></tr>
2720             <tr><th>AGE</th><td th:text="*{age}"></td></tr>
2721         </table>
2722     </body>
2723 </html>
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
2737 -Name : BootJdbcDemo
2738 -Type : Maven
2739 -Packaging : Jar
2740 -Java Version : 8
2741 -Language : Java
2742 -Group : com.example
2743 -Artifact : BootJdbcDemo
2744 -Version : 0.0.1-SNAPSHOT
2745 -Description : Demo project for Spring Boot
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
2757 -만일 Embedded된 tomcat으로 Stand-Alone형태로 구동시키기 위한 목적이라면 Packaging Type을 jar로
  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.
2760 -jar로 선택하고, local에서 개발 및 test를 하다가 나중에 배포할 경우 war로 변경해서 배포를 할 수도 있다.
2761
2762
2763 2. pom.xml
2764 1)jstl 사용을 위한
2765 <dependency>
2766 <groupId>javax.servlet</groupId>
2767 <artifactId>jstl</artifactId>
2768 <version>1.2</version>
2769 </dependency>
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>
2780 <groupId>org.mariadb.jdbc</groupId>
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
2792 spring.datasource.url=jdbc:mariadb://localhost:3306/test
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 (
2801 username VARCHAR(20) PRIMARY KEY,
2802 age TINYINT NOT NULL
```

```
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.NoArgsConstructor;
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 > Interface
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 {
2840         int create(UserVO userVO);
2841         List<UserVO> readAll();
2842         UserVO read(String username);
2843         int update(UserVO userVO);
2844         int delete(String username);
2845     }
2846
2847 5)com.example.dao > right-click > New > Class
2848 6)Name : UserDaoImpl
2849
2850     package com.example.dao;
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;
2858     import org.springframework.jdbc.core.RowMapper;
2859     import org.springframework.stereotype.Repository;
2860
```

```
2861 import com.example.vo.UserVO;
2862
2863 @Repository
2864 public class UserDaoImpl implements UserDao {
2865
2866     @Autowired
2867     private JdbcTemplate jdbcTemplate;
2868
2869     class UserMapper implements RowMapper<UserVO> {
2870         public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2871             UserVO user = new UserVO();
2872             user.setUsername(rs.getString("username"));
2873             user.setAge(rs.getInt("age"));
2874             return user;
2875         }
2876     }
2877
2878     @Override
2879     public int create(UserVO userVO) {
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
2890     @Override
2891     public UserVO read(String username) {
2892         String sql = "SELECT * FROM User WHERE username = ?";
2893         return this.jdbcTemplate.queryForObject(sql, new Object[] {username}, new
            UserMapper());
2894     }
2895
2896     @Override
2897     public int update(UserVO userVO) {
2898         String sql = "UPDATE User SET age = ? WHERE username = ?";
2899         return this.jdbcTemplate.update(sql, userVO.getAge(), userVO.getUsername());
2900     }
2901
2902     @Override
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
2926 @Controller
2927 public class MainController {
2928     @Autowired
2929     private UserDao userDao;
2930
2931     @GetMapping("/")
2932     public String index() {
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
2942     @GetMapping("/user")
2943     public String list(Model model) {
2944         model.addAttribute("users", this.userDao.readAll());
2945         return "list";
2946     }
2947 }
```

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 }
2964 h1{
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.
2977 8)folder를 다 만들었으면, WEB 루트(Context Root)로 인식할 있도록 환경설정을 아래와 같이 추가한다.
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"
2987     pageEncoding="UTF-8"%>
2988     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2989     <!DOCTYPE html>
2990     <html>
2991     <head>
2992     <meta charset="UTF-8">
2993     <title>New User Insertion Page</title>
2994     <link rel="stylesheet" type="text/css" href="/css/style.css">
2995     <c:url var="jsurl" value="/js/jquery-3.3.1.slim.min.js" />
2996     <script src="${jsurl}"></script>
2997     <script>
2998     $(document).ready(function() {
2999     alert("Hello, Spring Boot!");
3000     });
3001     </script>
3002     </head>
3003     <body>
3004     
3005     <h1>New User Insertion Page</h1>
3006     <form action="/user" method="post">
3007     Name : <input type="text" name="username" /><br />
3008     Age : <input type="number" name="age" /><br />
3009     <button type="submit">Submit</button>
3010     </form>
3011     </body>
3012     </html>
3013
3014 2)list.jsp
3015     <%@ page language="java" contentType="text/html; charset=UTF-8"
3016     pageEncoding="UTF-8"%>
3017     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3018     <!DOCTYPE html>
3019     <html>
3020     <head>
3021     <meta charset="UTF-8">
3022     <title>User List</title>
3023     <link rel="stylesheet" type="text/css" href="/css/style.css">
3024     </head>
3025     <body>
3026     <h1>User List</h1>
3027     <ul>
3028     <c:forEach var="user" items="${users}">
3029     <li>${user.username}(${user.age})</li>
3030     </c:forEach>
3031     </ul>
3032     </body>
3033     </html>
```

```

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
3043 @ComponentScan("com.example") <-- 추가 code
3044 public class BootJdbcDemoApplication {
3045
3046     public static void main(String[] args) {
3047         SpringApplication.run(BootJdbcDemoApplication.class, args);
3048     }
3049 }
3050
3051 -@SpringBootApplication 은 다음의 annotation 3개를 모두 담고 있다.
3052     --@SpringBootApplication
3053     --@EnableAutoConfiguration
3054     --@ComponentScan
3055 -만약, AutoScan이 되어야 하는 component class들 - 대표적으로 @Controller, @Service,
    @Repository, @Component 등-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
3061 15)http://localhost:8080
3062
3063
3064
3065 -----
3066 Task13. String Jdbc API 사용하기
3067 1. Spring Boot project 생성
3068     1)Package Explorer > right-click > New > Spring Starter Project
3069     2)다음의 각 항목의 값을 입력후 Next 클릭
3070         -Service URL :http://start.spring.io
3071         -Name : SpringBootJdbcDemo
3072         -Type : Maven
3073         -Packaging : Jar
3074         -Java Version : 8
3075         -Language : Java
3076         -Group : com.example
3077         -Artifact : SpringBootJdbcDemo
3078         -Version : 0.0.1-SNAPSHOT
3079         -Description : Demo project for Spring Boot
3080         -Package : com.example.biz
3081
3082     3)다음 각 항목을 선택후 Finish 클릭
3083         -Spring Boot Version : 2.2.6
3084         -Select
3085             --SQL > check Oracle Driver, JDBC API
3086             --Web > Spring Web

```



```
3087      --Developer Tools > Spring Boot DevTools, Lombok
3088      --Template Engines > Thymeleaf
3089      -Finish
3090
3091
3092 2. src/main/resources/application.properties
3093     spring.datasource.url=jdbc:mariadb://localhost:3306/test
3094     spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
3095     spring.datasource.username=root
3096     spring.datasource.password=javamariadb
3097
3098
3099 3. Table 생성
3100     CREATE TABLE Member
3101     (
3102         userid    VARCHAR2(20) PRIMARY KEY,
3103         username  VARCHAR2(20) NOT NULL,
3104         age       NUMBER(2) NOT NULL
3105     );
3106
3107
3108 4. VO object 생성
3109     1)src/main/java > right-click > New > Package
3110     2)Name : com.example.vo
3111     3)Finish
3112     4)com.example.vo > right-click > New > Class
3113     5)Name : MemberVO
3114
3115     package com.example.vo;
3116
3117     import lombok.AllArgsConstructor;
3118     import lombok.Getter;
3119     import lombok.NoArgsConstructor;
3120     import lombok.Setter;
3121     import lombok.ToString;
3122
3123     @Setter
3124     @Getter
3125     @NoArgsConstructor
3126     @AllArgsConstructor
3127     @ToString
3128     public class MemberVO {
3129         private String userid;
3130         private String username;
3131         private int age;
3132     }
3133
3134
3135 5. Dao object 생성
3136     1)src/main/java > right-click > New > Package
3137     2)Name : com.example.dao
3138     3)Finish
3139     4)com.example.dao > right-click > New > Interface
3140     5)Name : MemberDao
3141
3142     package com.example.dao;
3143
3144     import java.util.List;
```

```
3145     import com.example.vo.MemberVO;
3146
3147     public interface MemberDao {
3148         int create(MemberVO member);
3149
3150         List<MemberVO> readAll();
3151
3152         MemberVO read(String userid);
3153
3154         int update(MemberVO member);
3155
3156         int delete(String userid);
3157     }
3158
3159 6)com.example.dao > right-click > New > Class
3160 7)Name : MemberDaoImpl
3161 8)Interfaces : com.example.MemberDao
3162 9)Finish
3163
3164     package com.example.dao;
3165
3166     import java.sql.ResultSet;
3167     import java.sql.SQLException;
3168     import java.util.List;
3169
3170     import org.springframework.beans.factory.annotation.Autowired;
3171     import org.springframework.jdbc.core.JdbcTemplate;
3172     import org.springframework.jdbc.core.RowMapper;
3173     import org.springframework.stereotype.Repository;
3174
3175     import com.example.vo.MemberVO;
3176
3177     @Repository
3178     public class MemberDaoImpl implements MemberDao {
3179         @Autowired
3180         private JdbcTemplate jdbcTemplate;
3181
3182         @Override
3183         public int create(MemberVO member) {
3184             String sql = "INSERT INTO Member(userid, username, age) VALUES(?,?,?)";
3185             return this.jdbcTemplate.update(sql, member.getUserid(), member.getUsername(),
3186                 member.getAge());
3187         }
3188
3189         private class MyRowMapper implements RowMapper<MemberVO> {
3190             @Override
3191             public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
3192                 return new MemberVO(rs.getString("userid"), rs.getString("username"),
3193                     rs.getInt("age"));
3194             }
3195         }
3196
3197         @Override
3198         public List<MemberVO> readAll() {
3199             return this.jdbcTemplate.query("SELECT * FROM Member ORDER BY userid DESC",
3200                 new MyRowMapper());
3201         }
3202     }
```

```

3200     @Override
3201     public MemberVO read(String userid) {
3202         String sql = "SELECT * FROM Member WHERE userid = ?";
3203         return this.jdbcTemplate.queryForObject(sql, new Object[] { userid }, new
            MyRowMapper());
3204     }
3205
3206     @Override
3207     public int update(MemberVO member) {
3208         String sql = "UPDATE Member SET age = ?, username = ? WHERE userid = ?";
3209         return this.jdbcTemplate.update(sql, member.getAge(), member.getUsername(),
            member.getUserid());
3210     }
3211
3212     @Override
3213     public int delete(String userid) {
3214         String sql = "DELETE FROM Member WHERE userid = ?";
3215         return this.jdbcTemplate.update(sql, userid);
3216     }
3217 }

```

```

3218
3219

```

3220 6. Controller

```

3221 1)com.example.biz > right-click > New > Class
3222 2)Name : MainController
3223
3224     package com.example.biz;
3225
3226     import org.springframework.beans.factory.annotation.Autowired;
3227     import org.springframework.stereotype.Controller;
3228     import org.springframework.ui.Model;
3229     import org.springframework.web.bind.annotation.GetMapping;
3230     import org.springframework.web.bind.annotation.PostMapping;
3231
3232     import com.example.dao.MemberDao;
3233     import com.example.vo.MemberVO;
3234
3235     @Controller
3236     public class MainController {
3237         @Autowired
3238         private MemberDao memberDao;
3239
3240         @GetMapping("/")
3241         public String index() {
3242             return "index"; // templates/index.html
3243         }
3244
3245         @PostMapping("/member")
3246         public String insert(MemberVO member) {
3247             this.memberDao.create(member);
3248             return "redirect:/member";
3249         }
3250
3251         @GetMapping("/member")
3252         public String list(Model model) {
3253             model.addAttribute("members", this.memberDao.readAll());
3254             return "list"; // templates/list.html
3255         }

```

```
3256     }
3257
3258
3259 7. static resources 준비
3260 1)src/main/resources/static/images folder 생성
3261   -spring-boot.png 추가할 것
3262 2)src/main/resources/static/js folder 생성
3263   -jquery-3.5.0.min.js 추가할 것
3264 3)src/main/resources/static/css folder 생성
3265   -style.css
3266
3267     @charset "UTF-8";
3268     body {
3269         background-color:yellow;
3270     }
3271     h1{
3272         color : blue;
3273     }
3274
3275
3276 8. templates/index.html 생성
3277
3278 <!DOCTYPE html>
3279 <html>
3280 <head>
3281 <meta charset="UTF-8" />
3282 <title>New User Insertion Page</title>
3283 <link rel="stylesheet" type="text/css" href="/css/style.css">
3284 <script src="/js/jquery-3.5.0.min.js"></script>
3285 <script>
3286     $(document).ready(function() {
3287         alert("Hello, Spring Boot!");
3288     });
3289 </script>
3290 </head>
3291 <body>
3292     
3293     <h1>New User Insertion Page</h1>
3294     <form action="/member" method="post">
3295         <ul>
3296             <li>ID : <input type="text" name="userid" /></li>
3297             <li>Name : <input type="text" name="username" /></li>
3298             <li>Age : <input type="number" name="age" /></li>
3299             <li><button>Submit</button></li>
3300         </ul>
3301     </form>
3302 </body>
3303 </html>
3304
3305
3306 9. templates/list.html 생성
3307
3308 <!DOCTYPE html>
3309 <html xmlns:th="http://www.thymeleaf.org">
3310 <head>
3311 <meta charset="UTF-8" />
3312 <title>회원정보</title>
3313 <style>
```

```

3314     h1 {
3315         font-size: 18pt; font-weight: bold; color: gray;
3316     }
3317
3318     body {
3319         font-size: 13pt;   color: gray;   margin: 5px 25px;
3320     }
3321
3322     tr {
3323         margin: 5px;
3324     }
3325
3326     th {
3327         padding: 5px; color: white; background: darkgray;
3328     }
3329
3330     td {
3331         padding: 5px; color: black; background: #e0e0ff;
3332     }
3333 </style>
3334 </head>
3335 <body>
3336     <h1>회원명부</h1>
3337     <table border='1'>
3338         <thead>
3339             <tr>
3340                 <th>아이디</th>
3341                 <th>이름</th>
3342                 <th>나이</th>
3343             </tr>
3344         </thead>
3345         <tbody>
3346             <tr th:each="member : ${members}">
3347                 <td th:text="${member.userid}"></td>
3348                 <td th:text="${member.username}"></td>
3349                 <td th:text="${member.age}"></td>
3350             </tr>
3351         </tbody>
3352     </table>
3353 </body>
3354 </html>
3355
3356
3357 10. src/main/java/com.example.biz/SpringBootJdbcDemoApplication.java
3358
3359 package com.example.biz;
3360
3361 import org.springframework.boot.SpringApplication;
3362 import org.springframework.boot.autoconfigure.SpringBootApplication;
3363 import org.springframework.context.annotation.ComponentScan;
3364
3365 @SpringBootApplication
3366 @ComponentScan("com.example")
3367 public class SpringBootJdbcDemoApplication {
3368
3369     public static void main(String[] args) {
3370         SpringApplication.run(SpringBootJdbcDemoApplication.class, args);
3371     }

```

```

3372     }
3373
3374
3375 11. project > right-click > Run As > Spring Boot App
3376 12. http://localhost:8080
3377
3378
3379
3380 -----
3381 Task14. Spring Boot에서 MyBatis 사용하기
3382 1. Spring Boot project 생성
3383     1)Package Explorer > right-click > New > Spring Starter Project
3384     2)다음의 각 항목의 값을 입력후 Next 클릭
3385         -Service URL :http://start.spring.io
3386         -Name : SpringBootMembership
3387         -Type : Maven
3388         -Packaging : Jar
3389         -Java Version : 8
3390         -Language : Java
3391         -Group : com.example
3392         -Artifact : SpringBootMembership
3393         -Version : 0.0.1-SNAPSHOT
3394         -Description : Demo project for Spring Boot
3395         -Package : com.example.biz
3396
3397     3)다음 각 항목을 선택후 Finish 클릭
3398         -Spring Boot Version : 2.2.6
3399         -Select
3400             --Developer Tools > Spring Boot DevTools, Lombok, Spring Configuration Processor
3401             --SQL > check Oracle Driver, MyBatis Framework
3402             --Template Engines > Thymeleaf
3403             --Web > Spring Web
3404         -Finish
3405
3406
3407 2. membership.sql file 만들기
3408     1)src/main/resources > New > File
3409     2)File name : membership.sql
3410
3411     ALTER SESSION SET "_ORACLE_SCRIPT"=true;
3412
3413     CREATE USER membership IDENTIFIED BY membership
3414     DEFAULT TABLESPACE USERS
3415     TEMPORARY TABLESPACE TEMP;
3416
3417     ALTER USER membership
3418     DEFAULT TABLESPACE USERS
3419     QUOTA UNLIMITED ON USERS;
3420
3421     GRANT resource, connect TO membership;
3422
3423     conn membershop/membership
3424
3425     CREATE TABLE Member
3426     (
3427         userid      VARCHAR2(20),
3428         passwd      VARCHAR2(20) CONSTRAINT member_passwd_nn NOT NULL,
3429         name        VARCHAR2(20) CONSTRAINT member_name_nn  NOT NULL,

```

```

3430     age          NUMBER(2)    CONSTRAINT member_age_nn    NOT NULL,
3431     gender        CHAR(1)      CONSTRAINT member_gender_nn NOT NULL,
3432     city          VARCHAR2(30) CONSTRAINT member_city_nn   NOT NULL,
3433     CONSTRAINT    member_userid_pk PRIMARY KEY(userid),
3434     CONSTRAINT    member_age_ck CHECK(age BETWEEN 20 AND 65),
3435     CONSTRAINT    member_gender_ck CHECK(gender IN ('1', '0')),
3436     CONSTRAINT    member_city_ck CHECK(city IN('서울', '부산', '대전', '광주', '대구'))
3437 )

```

```

3438
3439 TRUNCATE TABLE Member;

```

```

3440

```

```

3441

```

```

3442 3. src/main/resources/application.properties

```

```

3443 spring.datasource.url=jdbc:oracle:thin:@localhost:1521:XE <--반드시 넣을 것, 매우 주의

```

```

3444 spring.datasource.hikari.driver-class-name=oracle.jdbc.driver.OracleDriver

```

```

3445 spring.datasource.hikari.jdbc-url=jdbc:oracle:thin:@localhost:1521:XE

```

```

3446 spring.datasource.hikari.username=membership

```

```

3447 spring.datasource.hikari.password=membership

```

```

3448 spring.datasource.hikari.connection-timeout=20000

```

```

3449 spring.datasource.hikari.minimum-idle=5

```

```

3450 spring.datasource.hikari.maximum-pool-size=20

```

```

3451 spring.datasource.hikari.idle-timeout=300000

```

```

3452 spring.datasource.hikari.max-lifetime=1200000

```

```

3453 spring.datasource.hikari.auto-commit=true

```

```

3454

```

```

3455

```

```

3456 4. DataSource 설정하기

```

```

3457 1)src/main/java > right-click > New > package

```

```

3458 2)Name : com.example.config

```

```

3459 3)Finish

```

```

3460 4)com.example.config > right-click > New > Class

```

```

3461 5)Name : DatabaseConfiguration

```

```

3462 6)Finish

```

```

3463

```

```

3464     package com.example.config;

```

```

3465

```

```

3466     import javax.sql.DataSource;

```

```

3467

```

```

3468     import org.springframework.boot.context.properties.ConfigurationProperties;

```

```

3469     import org.springframework.context.annotation.Bean;

```

```

3470     import org.springframework.context.annotation.Configuration;

```

```

3471     import org.springframework.context.annotation.PropertySource;

```

```

3472

```

```

3473     import com.zaxxer.hikari.HikariConfig;

```

```

3474     import com.zaxxer.hikari.HikariDataSource;

```

```

3475

```

```

3476     import lombok.extern.slf4j.Slf4j;

```

```

3477

```

```

3478     @Slf4j

```

```

3479     @Configuration

```

```

3480     @PropertySource("classpath:application.properties")

```

```

3481     public class DatabaseConfiguration {

```

```

3482

```

```

3483         @Bean

```

```

3484         @ConfigurationProperties(prefix="spring.datasource.hikari")

```

```

3485         public HikariConfig hikariConfig() {

```

```

3486             return new HikariConfig();

```

```

3487         }

```

```

3488
3489     @Bean
3490     public DataSource dataSource() throws Exception{
3491         DataSource dataSource = new HikariDataSource(hikariConfig());
3492         log.info("datasource : {}", dataSource);
3493         return dataSource;
3494     }
3495 }
3496
3497
3498 5. src/main/java/com.example.biz.SpringBootMembershipApplication
3499
3500 package com.example.biz;
3501
3502 import org.springframework.boot.SpringApplication;
3503 import org.springframework.boot.autoconfigure.SpringBootApplication;
3504 import org.springframework.context.annotation.ComponentScan;
3505
3506 @SpringBootApplication
3507 @ComponentScan(basePackages = "com.example")
3508 public class SpringBootMembershipApplication {
3509
3510     public static void main(String[] args) {
3511         SpringApplication.run(SpringBootMembershipApplication.class, args);
3512     }
3513 }
3514
3515
3516 6. Connection Test
3517 1)이제까지 설정한 것은 hikariConfig에서 설정한 정보를 가지고 DataSoure를 생성한 것이다.
3518 2)설정이 제대로 된 것인지 Project를 실행해 보자.
3519 3)logger에 의해 dataSource 정보를 확인할 수 있다.
3520
3521 4)Project > Run As > Spring Boot App
3522
3523 2020-04-21 23:50:38.033 INFO 8420 --- [ restartedMain]
3524 com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
3525 2020-04-21 23:50:38.037 WARN 8420 --- [ restartedMain]
3526 com.zaxxer.hikari.util.DriverDataSource : Registered driver with
3527 driverClassName=oracle.jdbc.driver.OracleDriver was not found, trying direct
3528 instantiation.
3529 2020-04-21 23:50:38.252 INFO 8420 --- [ restartedMain]
3530 com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
3531 2020-04-21 23:50:38.252 INFO 8420 --- [ restartedMain]
3532 c.example.config.DatabaseConfiguration : datasource : HikariDataSource (HikariPool-1)
3533
3534
3535 7. Mbatis 연동하기
3536 1)Mapper Folder 생성하기
3537 -Mapper Folder에는 Application에서 사용할 Query를 담고 있는 XML 파일을 저장한다.
3538 -src/main/resources/mapper folder 생성한다.
3539 -src/main/resources > right-click > New > Folder
3540 -Folder name : mapper
3541 -Finish
3542
3543 2)src/main/resources/mapper/mybatis-config.xml 생성
3544
3545 <?xml version="1.0" encoding="UTF-8"?>

```



```
3540 <!DOCTYPE configuration
3541 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
3542 "http://mybatis.org/dtd/mybatis-3-config.dtd">
3543 <configuration>
3544 <typeAliases>
3545 <typeAlias type="com.example.vo.MemberVO" alias="memberVO"/>
3546 </typeAliases>
3547 </configuration>
3548
3549 3)src/main/resources/mapper/mybatis-mapper.xml 생성
3550
3551 <?xml version="1.0" encoding="UTF-8"?>
3552 <!DOCTYPE mapper
3553 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3554 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
3555 <mapper namespace="com.example.vo.MemberVO">
3556 </mapper>
3557
3558 4)com.example.config.DatabaseConfiguration.java 수정하기
3559
3560 package com.example.config;
3561
3562 import javax.sql.DataSource;
3563
3564 import org.apache.ibatis.session.SqlSessionFactory;
3565 import org.mybatis.spring.SqlSessionFactoryBean;
3566 import org.mybatis.spring.SqlSessionTemplate;
3567 import org.springframework.beans.factory.annotation.Autowired;
3568 import org.springframework.boot.context.properties.ConfigurationProperties;
3569 import org.springframework.context.ApplicationContext;
3570 import org.springframework.context.annotation.Bean;
3571 import org.springframework.context.annotation.Configuration;
3572 import org.springframework.context.annotation.PropertySource;
3573 import org.springframework.core.io.Resource;
3574 import org.springframework.core.io.support.PathMatchingResourcePatternResolver;
3575
3576 import com.zaxxer.hikari.HikariConfig;
3577 import com.zaxxer.hikari.HikariDataSource;
3578
3579 import lombok.extern.slf4j.Slf4j;
3580
3581 @Slf4j
3582 @Configuration
3583 @PropertySource("classpath:application.properties")
3584 public class DatabaseConfiguration {
3585     @Autowired
3586     private ApplicationContext applicationContext;
3587
3588     @Bean
3589     @ConfigurationProperties(prefix="spring.datasource.hikari")
3590     public HikariConfig hikariConfig() {
3591         return new HikariConfig();
3592     }
3593
3594     @Bean
3595     public DataSource dataSource() throws Exception{
3596         DataSource dataSource = new HikariDataSource(hikariConfig());
3597         log.info("datasource : {}", dataSource);
3598     }
3599 }
```

```

3598     return dataSource;
3599 }
3600
3601 @Bean
3602 public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception{
3603     SqlSessionFactoryBean sqlSessionFactory = new SqlSessionFactoryBean();
3604     sqlSessionFactory.setDataSource(dataSource);
3605     Resource configLocation = new
        PathMatchingResourcePatternResolver().getResource("classpath:/mapper/mybatis-co
        nfig.xml");
3606     sqlSessionFactory.setMapperLocations(applicationContext.getResources("classpath:/
        mapper/mybatis-mapper.xml"));
3607     sqlSessionFactory.setConfigLocation(configLocation);
3608     return sqlSessionFactory.getObject();
3609 }
3610
3611 @Bean
3612 public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactory) {
3613     return new SqlSessionTemplate(sqlSessionFactory);
3614 }
3615 }
3616
3617
3618 8. vo, dao, service package 생성하기
3619 1)src/main/java > right-click > New > Package
3620 2)Name : com.example.vo
3621 3)src/main/java > right-click > New > Package
3622 4)Name : com.example.dao
3623 5)src/main/java > right-click > New > Package
3624 6)Name : com.example.service
3625
3626
3627 9. MemberVO class 생성하기
3628 1)com.example.vo > right-click > New > Class
3629 2)Name : MemberVO
3630 3)Finish
3631
3632 package com.example.vo;
3633
3634 import lombok.AllArgsConstructor;
3635 import lombok.Getter;
3636 import lombok.NoArgsConstructor;
3637 import lombok.Setter;
3638 import lombok.ToString;
3639
3640 @Getter
3641 @Setter
3642 @ToString
3643 @NoArgsConstructor
3644 @AllArgsConstructor
3645 public class MemberVO {
3646     private String userid;
3647     private String passwd;
3648     private String name;
3649     private int age;
3650     private String gender;
3651     private String city;
3652 }

```

```

3653
3654
3655 10. MyBatis 연결 Test
3656     -src/test/java/com.example.biz.SpringBootMembershipApplicationTests.java
3657
3658     package com.example.biz;
3659
3660     import org.junit.jupiter.api.Test;
3661     import org.mybatis.spring.SqlSessionTemplate;
3662     import org.springframework.beans.factory.annotation.Autowired;
3663     import org.springframework.boot.test.context.SpringBootTest;
3664
3665     import lombok.extern.slf4j.Slf4j;
3666
3667     @Slf4j
3668     @SpringBootTest
3669     class SpringBootMembershipApplicationTests {
3670         @Autowired
3671         private SqlSessionTemplate sqlSession;
3672
3673         @Test
3674         void contextLoads() {
3675         }
3676
3677         @Test
3678         public void test1() throws Exception{
3679             log.info("SqlSession = " + this.sqlSession);
3680         }
3681     }
3682
3683     -Run As > JUnit Test
3684     Green Bar
3685     2020-04-22 21:45:04.154 INFO 9628 --- [           main]
3686     e.b.SpringBootMembershipApplicationTests : Started
3687     SpringBootMembershipApplicationTests in 5.494 seconds (JVM running for 8.844)
3688     2020-04-22 21:45:04.702 INFO 9628 --- [           main]
3689     e.b.SpringBootMembershipApplicationTests : SqlSession =
3690     org.mybatis.spring.SqlSessionTemplate@3f357c9d
3691
3692 11. MemberDao, MemberDaoImpl 생성하기
3693     1)com.example.dao > right-click > New > Interface
3694     2)Name : MemberDao
3695     3)Finish
3696
3697     package com.example.dao;
3698
3699     import java.util.Map;
3700
3701     import com.example.vo.MemberVO;
3702
3703     public interface MemberDao {
3704         void create(MemberVO member);
3705         void readAll(Map map);
3706         void read(Map map);
3707         void update(MemberVO member);
3708         void delete(String userid);
3709     }

```

```
3707
3708 4)com.example.dao > right-click > New > Class
3709 5)Name : MemberDaoImpl
3710 6)Interfaces : com.example.dao.MemberDao
3711 7)Finish
3712
3713
3714 12. MemberService, MemberServiceImpl 생성하기
3715 1)com.example.service > right-click > New > Interface
3716 2)Name : MemberService
3717 3)Finish
3718
3719 package com.example.service;
3720
3721 import java.util.Map;
3722
3723 import com.example.vo.MemberVO;
3724
3725 public interface MemberService {
3726     void insertMember(MemberVO member);
3727     void selectAllMembers(Map map);
3728     void selectMember(Map map);
3729     void updateMember(MemberVO member);
3730     void deleteMember(String userid);
3731 }
3732
3733 4)com.example.service > right-click > New > Class
3734 5)Name : MemberServiceImpl
3735 6)Interfaces : com.example.service.MemberService
3736 7)Finish
3737
3738
3739 13. css, images, js folder 생성하기
3740 1)src/main/resources/static > right-click > New > Folder
3741 2)Folder name : css
3742     -bootstrap-theme.min.css
3743     -bootstrap.min.css
3744
3745 3)src/main/resources/static > right-click > New > Folder
3746 4)Folder name : images
3747     -spring-boot.png
3748
3749 5)src/main/resources/static > right-click > New > Folder
3750 6)Folder name : js
3751     -bootstrap.min.js
3752     -jquery-3.5.0.min.js
3753
3754 7)src/main/resources/static > right-click > New > Folder
3755 8)Folder name : fonts
3756     -glyphicons font 5개
3757
3758
3759 14. src/main/resources/templates에 index.html 생성하기
3760
3761 <!DOCTYPE html>
3762 <html lang="en" xmlns:th="https://www.thymeleaf.org/">
3763 <head>
3764     <meta charset="UTF-8" />
```

```

3765     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
3766     <title>Welcome example.com</title>
3767     <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
3768     <link rel="stylesheet" th:href="@{/css/bootstrap-theme.min.css}" />
3769     <script th:src="@{/js/jquery-3.5.0.min.js}"></script>
3770 </head>
3771 <body>
3772     <div class="jumbotron">
3773         <h1>Welcome to www.example.com</h1>
3774         <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nemo accusantium,
        aspernatur quos porro commodi perspiciatis, assumenda consequuntur eius inventore
        eaque omnis, magni natus corrupti doloremque? Quia aliquid excepturi tempora
        praesentium modi necessitatibus sequi quisquam dolorem nihil deserunt magnam,
        distinctio sunt aspernatur nisi. Id odio quaerat amet quidem adipisci totam ad.</p>
3775         <p>
3776             <a class="btn btn-primary btn-lg" th:href="@{register}" role="button">회원가입</a>
3777         </p>
3778     </div>
3779 </body>
3780 </html>

```

3781

3782

3783 15. MainController.java 생성

3784 1)com.example.biz > right-click > New > Class

3785 2)Name : MainController

3786 3)Finish

3787

3788 package com.example.biz;

3789

3790 import org.springframework.stereotype.Controller;

3791 import org.springframework.web.bind.annotation.GetMapping;

3792

3793 @Controller

3794 public class MainController {

3795

3796 @GetMapping("/")

3797 public String index() {

3798 return "index";

3799 }

3800 }

3801

3802

3803 4)SpringBootMembership project > right-click > Run As > Spring Boot App

3804 5)<http://localhost:8080>

3805

3806

3807 16. 회원가입하기

3808 1)MainController.java 코드 추가

3809

3810 @GetMapping("/register")

3811 public String register() {

3812 return "register";

3813 }

3814

3815 2)templates > right-click > New > Web > HTML File

3816 3)Name : register.html

3817

3818 <!DOCTYPE html>

```

3819 <html lang="en" xmlns:th="https://www.thymeleaf.org/">
3820 <head>
3821   <meta charset="UTF-8" />
3822   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
3823   <title>회원가입</title>
3824   <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
3825   <script th:src="@{/js/jquery-3.5.0.min.js}"></script>
3826   <script>
3827     $(function () {
3828       var flag = 0;
3829       $("#userid").bind("change", function () {
3830         if ($("#userid").val()) {
3831           $("#useriddiv").attr("class", "form-group has-success has-feedback");
3832           $("#useridicon").attr("class", "glyphicon glyphicon-ok
3833             form-control-feedback");
3834           $("#useridsr").text("(success)");
3835           flag++;
3836         }
3837       });
3838       $("#passwd").bind("change", function () {
3839         if ($("#passwd").val()) {
3840           $("#passwddiv").attr("class", "form-group has-success has-feedback");
3841           $("#passwdicon").attr("class", "glyphicon glyphicon-ok
3842             form-control-feedback");
3843           $("#passwdsr").text("(success)");
3844           flag++;
3845         }
3846       });
3847       $("#name").bind("change", function () {
3848         if ($("#name").val()) {
3849           $("#namediv").attr("class", "form-group has-success has-feedback");
3850           $("#nameicon").attr("class", "glyphicon glyphicon-ok
3851             form-control-feedback");
3852           $("#namesr").text("(success)");
3853           flag++;
3854         }
3855       });
3856       $("#age").bind("keyup", function () {
3857         if ($("#age").val() >= 20 && ($("#age").val() <= 65) {
3858           $("#agediv").attr("class", "form-group has-success has-feedback");
3859           $("#ageicon").attr("class", "glyphicon glyphicon-ok
3860             form-control-feedback");
3861           $("#agesr").text("(success)");
3862           flag++;
3863         }
3864       });
3865       $(".:radio").bind("click", function () {
3866         $("#genderdiv").attr("class", "form-group has-success has-feedback");
3867         flag++;
3868       });
3869       $("#city").bind("change", function () {
3870         if ($("#city").val()) {
3871           $("#citydiv").attr("class", "form-group has-success has-feedback");
3872           flag++;
3873           if (flag == 6) {
3874             $('button').removeAttr("disabled");
3875           }
3876         }
3877       });
3878     });
3879   </script>
3880 </head>
3881 <body>
3882   <div class="container">
3883     <div class="row">
3884       <div class="col-md-12">
3885         <div class="text-align-center">
3886           <h2>회원가입</h2>
3887         </div>
3888         <div class="form">
3889           <div class="form-group">
3890             <input type="text" value="" class="form-control" />
3891             <div class="feedback" style="display: none;>
3892               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3893             </div>
3894             <input type="checkbox" value="" class="checkbox" />
3895             <div class="checkbox">
3896               <input type="checkbox" value="" class="checkbox" />
3897             </div>
3898           </div>
3899           <div class="form-group">
3900             <input type="password" value="" class="form-control" />
3901             <div class="feedback" style="display: none;>
3902               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3903             </div>
3904           </div>
3905           <div class="form-group">
3906             <input type="text" value="" class="form-control" />
3907             <div class="feedback" style="display: none;>
3908               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3909             </div>
3910           </div>
3911           <div class="form-group">
3912             <input type="text" value="" class="form-control" />
3913             <div class="feedback" style="display: none;>
3914               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3915             </div>
3916           </div>
3917           <div class="form-group">
3918             <input type="text" value="" class="form-control" />
3919             <div class="feedback" style="display: none;>
3920               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3921             </div>
3922           </div>
3923           <div class="form-group">
3924             <input type="text" value="" class="form-control" />
3925             <div class="feedback" style="display: none;>
3926               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3927             </div>
3928           </div>
3929           <div class="form-group">
3930             <input type="text" value="" class="form-control" />
3931             <div class="feedback" style="display: none;>
3932               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3933             </div>
3934           </div>
3935           <div class="form-group">
3936             <input type="text" value="" class="form-control" />
3937             <div class="feedback" style="display: none;>
3938               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3939             </div>
3940           </div>
3941           <div class="form-group">
3942             <input type="text" value="" class="form-control" />
3943             <div class="feedback" style="display: none;>
3944               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3945             </div>
3946           </div>
3947           <div class="form-group">
3948             <input type="text" value="" class="form-control" />
3949             <div class="feedback" style="display: none;>
3950               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3951             </div>
3952           </div>
3953           <div class="form-group">
3954             <input type="text" value="" class="form-control" />
3955             <div class="feedback" style="display: none;>
3956               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3957             </div>
3958           </div>
3959           <div class="form-group">
3960             <input type="text" value="" class="form-control" />
3961             <div class="feedback" style="display: none;>
3962               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3963             </div>
3964           </div>
3965           <div class="form-group">
3966             <input type="text" value="" class="form-control" />
3967             <div class="feedback" style="display: none;>
3968               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3969             </div>
3970           </div>
3971           <div class="form-group">
3972             <input type="text" value="" class="form-control" />
3973             <div class="feedback" style="display: none;>
3974               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3975             </div>
3976           </div>
3977           <div class="form-group">
3978             <input type="text" value="" class="form-control" />
3979             <div class="feedback" style="display: none;>
3980               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3981             </div>
3982           </div>
3983           <div class="form-group">
3984             <input type="text" value="" class="form-control" />
3985             <div class="feedback" style="display: none;>
3986               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3987             </div>
3988           </div>
3989           <div class="form-group">
3990             <input type="text" value="" class="form-control" />
3991             <div class="feedback" style="display: none;>
3992               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3993             </div>
3994           </div>
3995           <div class="form-group">
3996             <input type="text" value="" class="form-control" />
3997             <div class="feedback" style="display: none;>
3998               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
3999             </div>
4000           </div>
4001           <div class="form-group">
4002             <input type="text" value="" class="form-control" />
4003             <div class="feedback" style="display: none;>
4004               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
4005             </div>
4006           </div>
4007           <div class="form-group">
4008             <input type="text" value="" class="form-control" />
4009             <div class="feedback" style="display: none;>
4010               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
4011             </div>
4012           </div>
4013           <div class="form-group">
4014             <input type="text" value="" class="form-control" />
4015             <div class="feedback" style="display: none;>
4016               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
4017             </div>
4018           </div>
4019           <div class="form-group">
4020             <input type="text" value="" class="form-control" />
4021             <div class="feedback" style="display: none;>
4022               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
4023             </div>
4024           </div>
4025           <div class="form-group">
4026             <input type="text" value="" class="form-control" />
4027             <div class="feedback" style="display: none;>
4028               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
4029             </div>
4030           </div>
4031           <div class="form-group">
4032             <input type="text" value="" class="form-control" />
4033             <div class="feedback" style="display: none;>
4034               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
4035             </div>
4036           </div>
4037           <div class="form-group">
4038             <input type="text" value="" class="form-control" />
4039             <div class="feedback" style="display: none;>
4040               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
4041             </div>
4042           </div>
4043           <div class="form-group">
4044             <input type="text" value="" class="form-control" />
4045             <div class="feedback" style="display: none;>
4046               <div class="glyphicon glyphicon-ok form-control-feedback"></div>
4047             </div>
4048           </div>
4049           <div class="form-group">
4050             <input type="text" value="" class="form-control" /
```

```

3873     });
3874     })
3875     </script>
3876 </head>
3877 <body>
3878     <div class="container">
3879         <div class="row">
3880             <h1 class="h1 text-center">회원가입</h1>
3881         </div>
3882         <div class="row">
3883             <form class="form-horizontal" th:action="@{/register}" method="post">
3884                 <div id="useriddiv" class="form-group has-error has-feedback">
3885                     <label for="userid" class="col-sm-3 control-label">User ID</label>
3886                     <div class="col-sm-6">
3887                         <div class="input-group">
3888                             <span class="input-group-addon"><span class="glyphicon
3889 glyphicon-trash"></span></span>
3890                             <input type="text" name="userid" id="userid" class="form-control"
3891                                 placeholder="Enter your ID">
3892                         </div>
3893                         <span id="useridicon" class="glyphicon glyphicon-remove
3894                             form-control-feedback"
3895                             aria-hidden="true"></span>
3896                         <span id="useridsr" class="sr-only">(error)</span>
3897                     </div>
3898                 </div>
3899                 <div id="passwddiv" class="form-group has-error has-feedback">
3900                     <label for="passwd" class="col-sm-3 control-label">Password</label>
3901                     <div class="col-sm-6">
3902                         <div class="input-group">
3903                             <span class="input-group-addon"><span class="glyphicon
3904 glyphicon-lock"></span></span>
3905                             <input type="password" name="passwd" id="passwd"
3906                                 class="form-control"
3907                                 placeholder="Enter your password">
3908                         </div>
3909                         <span id="passwdicon" class="glyphicon glyphicon-remove
3910                             form-control-feedback"
3911                             aria-hidden="true"></span>
3912                         <span id="passwdsr" class="sr-only">(error)</span>
3913                     </div>
3914                 </div>
3915                 <div id="namediv" class="form-group has-error has-feedback">
3916                     <label for="name" class="col-sm-3 control-label">Name</label>
3917                     <div class="col-sm-6">
3918                         <div class="input-group">
3919                             <span class="input-group-addon"><span class="glyphicon
3920 glyphicon-user"></span></span>
3921                             <input type="text" name="name" id="name" class="form-control"
3922                                 placeholder="Enter your name">
3923                         </div>
3924                         <span id="nameicon" class="glyphicon glyphicon-remove
3925                             form-control-feedback"
3926                             aria-hidden="true"></span>
3927                         <span id="namesr" class="sr-only">(error)</span>
3928                     </div>
3929                 </div>
3930                 <div id="agediv" class="form-group has-error has-feedback">

```

```

3926         <label for="age" class="col-sm-3 control-label">Age</label>
3927         <div class="col-sm-4">
3928             <div class="input-group">
3929                 <span class="input-group-addon"><span class="glyphicon
3930 glyphicon-music"></span></span>
3931                 <input type="number" name="age" id="age" class="form-control"
                 placeholder="Enter your age">
3932             </div>
3933             <span id="ageicon" class="glyphicon glyphicon-remove
             form-control-feedback"
3934             aria-hidden="true"></span>
3935             <span id="agesr" class="sr-only">(error)</span>
3936         </div>
3937     </div>
3938     <div id="genderdiv" class="form-group has-error has-feedback">
3939         <label class="col-sm-3 control-label">Gender</label>
3940         <div class="col-sm-6">
3941             <label class="radio-inline">
3942                 <input type="radio" name="gender" value="1">Male
3943             </label>
3944             <label class="radio-inline">
3945                 <input type="radio" name="gender" value="0">Female
3946             </label>
3947         </div>
3948     </div>
3949     <div id="citydiv" class="form-group has-error has-feedback">
3950         <label for="city" class="col-sm-3 control-label">City</label>
3951         <div class="col-sm-6">
3952             <select name="city" id="city" class="form-control">
3953                 <option value="">--선택--</option>
3954                 <option value="서울">서울</option>
3955                 <option value="부산">부산</option>
3956                 <option value="대전">대전</option>
3957                 <option value="광주">광주</option>
3958                 <option value="대구">대구</option>
3959             </select>
3960         </div>
3961     </div>
3962     <div class="form-group">
3963         <div class="col-sm-6 col-sm-offset-3">
3964             <button class="col-sm-4 btn btn-primary" disabled>Submit</button>
3965         </div>
3966     </div>
3967 </form>
3968 </div>
3969 </div>
3970 </body>
3971 </html>

```

4) MainController.java 수정

```

3973 @Controller
3974 public class MainController {
3975
3976     @Autowired
3977     private MemberService memberService;
3978
3979     @GetMapping("/")
3980
3981

```



```
3982     public String index() {
3983         return "index";
3984     }
3985
3986     @GetMapping("/register")
3987     public String register() {
3988         return "register";
3989     }
3990
3991     @PostMapping("/register")
3992     public String register(MemberVO member) {
3993         this.memberService.insertMember(member);
3994         return "redirect:/";
3995     }
3996 }
3997
3998
3999 5)com.example.service.MemberServiceImpl.java 수정하기
4000
4001     package com.example.service;
4002
4003     import java.util.Map;
4004
4005     import org.springframework.beans.factory.annotation.Autowired;
4006     import org.springframework.stereotype.Service;
4007
4008     import com.example.dao.MemberDao;
4009     import com.example.vo.MemberVO;
4010
4011     @Service("memberService")
4012     public class MemberServiceImpl implements MemberService {
4013         @Autowired
4014         private MemberDao memberDao;
4015
4016         @Override
4017         public void insertMember(MemberVO member) {
4018             this.memberDao.create(member);
4019         }
4020
4021 6)com.example.dao.MemberDaoImpl.java 수정하기
4022
4023     package com.example.dao;
4024
4025     import java.util.Map;
4026
4027     import org.apache.ibatis.session.SqlSession;
4028     import org.springframework.beans.factory.annotation.Autowired;
4029     import org.springframework.stereotype.Repository;
4030
4031     import com.example.vo.MemberVO;
4032
4033     @Repository("memberDao")
4034     public class MemberDaoImpl implements MemberDao {
4035         @Autowired
4036         private SqlSession sqlSession;
4037
4038         @Override
4039         public void create(MemberVO member) {
```



```
4090     return "redirect:/list";
4091 }
4092
4093 @GetMapping("/list")
4094 public String list(Model model) {
4095     Map<String, Object> map = new HashMap<String, Object>();
4096     this.memberService.selectAllMembers(map);
4097     List<MemberVO> list = (List<MemberVO>)map.get("results");
4098     //list.forEach(member -> log.info("'" + member));
4099     model.addAttribute("members", list);
4100     return "list"; //templates/list.html
4101 }
4102
4103 3)MemberServiceImpl.java 수정하기
4104
4105 @Override
4106 public void selectAllMembers(Map map) {
4107     this.memberDao.readAll(map);
4108 }
4109
4110 4)MemberDaoImpl.java
4111
4112 @Override
4113 public void readAll(Map map) {
4114     this.sqlSession.selectList("selectAll", map);
4115 }
4116
4117 5)templates/list.html
4118
4119 <!DOCTYPE html>
4120 <html lang="en" xmlns:th="https://www.thymeleaf.org/">
4121 <head>
4122     <meta charset="UTF-8" />
4123     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
4124     <title>example.com 회원명단</title>
4125     <style type="text/css">
4126         body {
4127             margin:0px;
4128             padding:0px;
4129         }
4130         h1 { text-align: center;}
4131         table {
4132             width:500px;
4133             margin-left: auto;
4134             margin-right: auto;
4135             border-collapse: collapse;
4136             border : 1px solid white;
4137         }
4138         th, td {
4139             border :1px solid blue;
4140         }
4141         th{
4142             background-color: navy;
4143             color : white;
4144             height:40px;
4145         }
4146         td{
4147             text-align: center;
```

```

4148     }
4149     </style>
4150 </head>
4151 <body>
4152     <div>
4153         <a th:href="@{/}"></a>
4154     </div>
4155     <h1>회원명단</h1>
4156     <table>
4157         <thead>
4158             <tr>
4159                 <th>아이디</th>
4160                 <th>이름</th>
4161                 <th>나이</th>
4162                 <th>성별</th>
4163                 <th>거주지</th>
4164             </tr>
4165         </thead>
4166         <tbody>
4167             <tr th:each="member:${members}">
4168                 <td th:text="${member.userid}"></td>
4169                 <td th:text="${member.name}"></td>
4170                 <td th:text="${member.age}"></td>
4171                 <td th:if="${member.gender eq '1'}" th:text="남성"></td>
4172                 <td th:if="${member.gender eq '0'}" th:text="여성"></td>
4173                 <td th:text="${member.city}"></td>
4174             </tr>
4175         </tbody>
4176     </table>
4177 </body>
4178 </html>

```

6)mybatis-mapper.xml

```

4181
4182 <resultMap type="memberVO" id="selectMap">
4183     <result property="userid" column="userid"/>
4184     <result property="passwd" column="passwd"/>
4185     <result property="name" column="name"/>
4186     <result property="age" column="age"/>
4187     <result property="gender" column="gender"/>
4188     <result property="city" column="city"/>
4189 </resultMap>
4190
4191 <parameterMap type="hashmap" id="selectAllParameterMap">
4192     <parameter property="results" javaType="ResultSet" jdbcType="CURSOR"
4193         mode="OUT" resultMap="selectMap"/>
4194 </parameterMap>
4195
4196 <select id="selectAll" parameterMap="selectAllParameterMap"
4197     statementType="CALLABLE">
4198     { call member_select_all_sp(?) }
4199 </select>

```

7)member_select_all_sp

```

4200
4201 CREATE OR REPLACE PROCEDURE member_select_all_sp
4202 (
4203     member_record OUT SYS_REFCURSOR

```

```

4204     )
4205     AS
4206     BEGIN
4207         OPEN member_record FOR
4208         SELECT * FROM Member
4209         ORDER BY userid ASC;
4210     END;
4211
4212 8) http://localhost:8080/list
4213
4214
4215 18. 특정 회원 정보 가져오기
4216 1)list.html 수정하기
4217
4218     <tr th:each="member : ${members}">
4219         <td><a th:href="@{/member/{userid}(userid=${member.userid})}"
4220             th:text="${member.userid}"></a></td>
4221         <td th:text="${member.name}"></td>
4222         <td th:text="${member.age}"></td>
4223         <td th:if="${member.gender eq '1'}" th:text="남성"></td>
4224         <td th:if="${member.gender eq '0'}" th:text="여성"></td>
4225         <td th:text="${member.city}"></td>
4226     </tr>
4227
4228 2)MainController.java
4229
4230     @GetMapping("/member/{userid}")
4231     public String display(@PathVariable String userid, Model model) {
4232         Map<String, Object> map = new HashMap<String, Object>();
4233         map.put("userid", userid);
4234         this.memberService.selectMember(map);
4235         List<MemberVO> list = (List<MemberVO>)map.get("result");
4236         MemberVO member = list.get(0);
4237         model.addAttribute("member", member);
4238         //log.info("user = {}", member);
4239         return "display"; //templates/display.html
4240     }
4241
4242 3)MemberServiceImpl.java
4243
4244     @Override
4245     public void selectMember(Map map) {
4246         this.memberDao.read(map);
4247     }
4248
4249 4)MemberDaoImpl.java
4250
4251     @Override
4252     public void read(Map map) {
4253         this.sqlSession.selectOne("select", map);
4254     }
4255
4256 5)mybatis-mapper.xml
4257
4258     <parameterMap type="hashmap" id="selectParameterMap">
4259         <parameter property="userid" javaType="java.lang.String" jdbcType="VARCHAR"
4260             mode="IN"/>
4260         <parameter property="result" javaType="ResultSet" jdbcType="CURSOR" mode="OUT"

```

1)display.html 수정하기

```

4316     <a th:href="@{/member/update/{userid}(userid=${member.userid})}">정보수정하기</a>
4317
4318 2)MainController.java 추가
4319
4320     @GetMapping("/member/update/{userid}")
4321     public String update(@PathVariable String userid, Model model) {
4322         Map<String, Object> map = new HashMap<String, Object>();
4323         map.put("userid", userid);
4324         this.memberService.selectMember(map);
4325         List<MemberVO> list = (List<MemberVO>)map.get("result");
4326         MemberVO member = list.get(0);
4327         model.addAttribute("member", member);
4328         //log.info("수정할 UserID = " + userid);
4329         return "update"; //templates/update.html
4330     }
4331
4332     @RequestMapping(value = "/member", method=RequestMethod.POST)
4333     public String update(MemberVO member) {
4334         //log.info("수정할 멤버 = " + member);
4335         this.memberService.updateMember(member);
4336         return "redirect:/member/" + member.getUserid();
4337     }
4338
4339 3)MemberServiceImpl.java 수정
4340
4341     @Override
4342     public void updateMember(MemberVO member) {
4343         this.memberDao.update(member);
4344     }
4345
4346 4)MemberDaoImpl.java 수정
4347
4348     @Override
4349     public void update(MemberVO member) {
4350         this.sqlSession.update("update", member);
4351     }
4352
4353 5)mybatis-mapper.xml
4354
4355     <parameterMap type="memberVO" id="parameterUpdateMap">
4356         <parameter property="age" javaType="java.lang.Integer" jdbcType="INTEGER" />
4357         <parameter property="gender" javaType="java.lang.String" jdbcType="VARCHAR"/>
4358         <parameter property="city" javaType="java.lang.String" jdbcType="VARCHAR"/>
4359         <parameter property="userid" javaType="java.lang.String" jdbcType="VARCHAR"/>
4360     </parameterMap>
4361     <update id="update" parameterType="memberVO"
4362         parameterMap="parameterUpdateMap" statementType="CALLABLE">
4363         { call member_update_sp(?,?,?,?) }
4364     </update>
4365
4366 6)member_update_sp
4367
4368     CREATE OR REPLACE PROCEDURE member_update_sp
4369     (
4370         t_age          IN          member.age%TYPE,
4371         t_gender        IN          member.gender%TYPE,
4372         t_city          IN          member.city%TYPE,
4373         t_userid        IN          member.userid%TYPE

```

```

4374 )
4375 IS
4376 BEGIN
4377     UPDATE Member SET age = t_age, gender = t_gender, city = t_city
4378     WHERE userid = t_userid;
4379     COMMIT;
4380 END;
4381
4382 7)templates/update.html
4383
4384 <!DOCTYPE html>
4385 <html lang="en" xmlns:th="https://www.thymeleaf.org/">
4386 <head>
4387     <meta charset="UTF-8" />
4388     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
4389     <title>회원정보 수정창</title>
4390     <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
4391     <script th:src="@{/js/jquery-3.5.0.min.js}"></script>
4392     <script>
4393         $(function () {
4394             //var flag = 0;
4395             $("#userid").bind("change", function () {
4396                 if ($("#userid").val()) {
4397                     $("#useriddiv").attr("class", "form-group has-success has-feedback");
4398                     $("#useridicon").attr("class", "glyphicon glyphicon-ok
4399                     form-control-feedback");
4400                     $("#useridsr").text("(success)");
4401                     //flag++;
4402                 }
4403             });
4404             $("#passwd").bind("change", function () {
4405                 if ($("#passwd").val()) {
4406                     $("#passwddiv").attr("class", "form-group has-success has-feedback");
4407                     $("#passwdicon").attr("class", "glyphicon glyphicon-ok
4408                     form-control-feedback");
4409                     $("#passwdsr").text("(success)");
4410                     //flag++;
4411                 }
4412             });
4413             $("#name").bind("change", function () {
4414                 if ($("#name").val()) {
4415                     $("#namediv").attr("class", "form-group has-success has-feedback");
4416                     $("#nameicon").attr("class", "glyphicon glyphicon-ok
4417                     form-control-feedback");
4418                     $("#namesr").text("(success)");
4419                     //flag++;
4420                 }
4421             });
4422             $("#age").bind("keyup", function () {
4423                 if ($("#age").val() >= 20 && ($("#age").val() <= 65) {
4424                     $("#agediv").attr("class", "form-group has-success has-feedback");
4425                     $("#ageicon").attr("class", "glyphicon glyphicon-ok
4426                     form-control-feedback");
4427                     $("#agesr").text("(success)");
4428                     //flag++;
4429                 }
4430             });
4431             $("radio").bind("click", function () {

```



```

4428         $("#genderdiv").attr("class", "form-group has-success has-feedback");
4429         //flag++;
4430     });
4431     $("#city").bind("change", function () {
4432         if ($("#city").val()) {
4433             $("#citydiv").attr("class", "form-group has-success has-feedback");
4434             //flag++;
4435             //if (flag == 6) {
4436                 //    $('button').removeAttr("disabled");
4437             //}
4438         }
4439     });
4440 })
4441 </script>
4442 </head>
4443 <body>
4444     <div class="container" th:object="${member}">
4445         <div class="row">
4446             <h1 class="h1 text-center">회원정보수정</h1>
4447         </div>
4448         <div class="row">
4449             <form class="form-horizontal" th:action="@{/member}" method="post">
4450                 <div id="useriddiv" class="form-group has-error has-feedback">
4451                     <label for="userid" class="col-sm-3 control-label">User ID</label>
4452                     <div class="col-sm-6">
4453                         <div class="input-group">
4454                             <span class="input-group-addon"><span class="glyphicon
4455 glyphicon-trash"></span></span></div>
4456                             <input type="text" name="userid" id="userid" class="form-control"
4457                                 placeholder="Enter your ID" readonly="readonly"
4458                                 th:attrappend="value=${userid}">
4459                             <span id="useridicon" class="glyphicon glyphicon-remove
4460 form-control-feedback"
4461                                 aria-hidden="true"></span>
4462                             <span id="useridsr" class="sr-only">(error)</span>
4463                         </div>
4464                     <div id="passwddiv" class="form-group has-error has-feedback">
4465                         <label for="passwd" class="col-sm-3 control-label">Password</label>
4466                         <div class="col-sm-6">
4467                             <div class="input-group">
4468                                 <span class="input-group-addon"><span class="glyphicon
4469 glyphicon-lock"></span></span></div>
4470                                 <input type="password" name="passwd" id="passwd"
4471                                     class="form-control"
4472                                     placeholder="Enter your password">
4473                                 <span id="passwdicon" class="glyphicon glyphicon-remove
4474 form-control-feedback"
4475                                     aria-hidden="true"></span>
4476                                 <span id="passwdsr" class="sr-only">(error)</span>
4477                             </div>
4478                         <div id="namediv" class="form-group has-error has-feedback">
4479                             <label for="name" class="col-sm-3 control-label">Name</label>
4480                             <div class="col-sm-6">
4481                                 <div class="input-group">

```

```

4482         <span class="input-group-addon"><span class="glyphicon
4483 glyphicon-user"></span></span>
4484         <input type="text" name="name" id="name" class="form-control"
4485         placeholder="Enter your name" readonly="readonly"
4486         th:attrappend="value={name}">
4487     </div>
4488     <span id="nameicon" class="glyphicon glyphicon-remove
4489     form-control-feedback"
4490     aria-hidden="true"></span>
4491     <span id="namesr" class="sr-only">(error)</span>
4492 </div>
4493 <div id="agediv" class="form-group has-error has-feedback">
4494     <label for="age" class="col-sm-3 control-label">Age</label>
4495     <div class="col-sm-4">
4496         <div class="input-group">
4497             <span class="input-group-addon"><span class="glyphicon
4498 glyphicon-music"></span></span>
4499             <input type="number" name="age" id="age" class="form-control"
4500             placeholder="Enter your age" th:attrappend="value={age}">
4501         </div>
4502         <span id="ageicon" class="glyphicon glyphicon-remove
4503         form-control-feedback"
4504         aria-hidden="true"></span>
4505         <span id="agesr" class="sr-only">(error)</span>
4506     </div>
4507 </div>
4508 <div id="genderdiv" class="form-group has-error has-feedback">
4509     <label class="col-sm-3 control-label">Gender</label>
4510     <div class="col-sm-6">
4511         <label class="radio-inline">
4512             <input type="radio" name="gender" value="1"
4513             th:checked="{(gender eq '1')} ? 'checked' : ''">Male
4514         </label>
4515         <label class="radio-inline">
4516             <input type="radio" name="gender" value="0"
4517             th:checked="{(gender eq '0')} ? 'checked' : ''">Female
4518         </label>
4519     </div>
4520 </div>
4521 <div id="citydiv" class="form-group has-error has-feedback">
4522     <label for="city" class="col-sm-3 control-label">City</label>
4523     <div class="col-sm-6">
4524         <select name="city" id="city" class="form-control">
4525             <option value="">--선택--</option>
4526             <option value="서울" th:selected="{(city eq '서울')} ? 'selected' :
4527             false}">서울</option>
4528             <option value="부산" th:selected="{(city eq '부산')} ? 'selected' :
4529             false}">부산</option>
4530             <option value="대전" th:selected="{(city eq '대전')} ? 'selected' :
4531             false}">대전</option>
4532             <option value="광주" th:selected="{(city eq '광주')} ? 'selected' :
4533             false}">광주</option>
4534             <option value="대구" th:selected="{(city eq '대구')} ? 'selected' :
4535             false}">대구</option>
4536         </select>
4537     </div>
4538 </div>

```

```

4532         <div class="form-group">
4533             <div class="col-sm-6 col-sm-offset-3">
4534                 <button class="col-sm-4 btn btn-primary">수정하기</button>
4535             </div>
4536         </div>
4537     </form>
4538 </div>
4539 </div>
4540 </body>
4541 </html>
4542
4543 8)특정 회원 정보 수정
4544
4545
4546 20. 회원 삭제하기
4547   1)display.html 수정
4548
4549       <a href="#" th:href="@{/list}">목록으로</a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
4550       <a th:href="@{/member/update/{userid}(userid=${member.userid})}">정보수정하기
4551       </a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
4552       <a th:href="@{/delete/{userid}(userid=${member.userid})}">탈퇴하기</a>
4553
4554   2>MainController.java
4555
4556       @RequestMapping(value="/delete/{userid}", method=RequestMethod.GET)
4557       public String delete(@PathVariable String userid) {
4558           this.memberService.deleteMember(userid);
4559           return "redirect:/";
4560       }
4561
4562   3)MemberServiceImpl.java
4563
4564       @Override
4565       public void deleteMember(String userid) {
4566           this.memberDao.delete(userid);
4567       }
4568
4569   4)MemberDaoImpl.java
4570
4571       @Override
4572       public void delete(String userid) {
4573           this.sqlSession.delete("delete", userid);
4574       }
4575
4576   5)mybatis-mapper.xml
4577
4578       <delete id="delete" parameterType="java.lang.String" statementType="CALLABLE">
4579           { call member_delete_sp(#{userid}) }
4580       </delete>
4581
4582   6)member_delete_sp
4583
4584       CREATE OR REPLACE PROCEDURE member_delete_sp
4585       (
4586           t_userid          IN      member.userid%TYPE
4587       )
4588       IS
4589       BEGIN
```

```
4589      DELETE FROM Member
4590      WHERE userid = t_userid;
4591      COMMIT;
4592  END;
4593
4594
4595 21. jar를 war로 repackagging 하기
4596 1)jar를 war로 변경
4597    -pom.xml 수정
4598
4599      <groupId>com.example</groupId>
4600      <artifactId>SpringBootMembership</artifactId>
4601      <version>0.0.1-SNAPSHOT</version>
4602      <packaging>war</packaging>          <---삽입
4603      <name>SpringBootMembership</name>
4604      <description>Demo project for Spring Boot</description>
4605
4606 2)embedded tomcat을 WAS로 변경하기
4607    -pom.xml 추가
4608
4609      <dependency>
4610        <groupId>org.springframework.boot</groupId>
4611        <artifactId>spring-boot-starter-tomcat</artifactId>
4612        <scope>provided</scope>
4613      </dependency>
4614
4615 3)com.example.demo.SpringBootMembershipApplication.java 수정하기
4616
4617    package com.example.biz;
4618
4619    import org.springframework.boot.SpringApplication;
4620    import org.springframework.boot.autoconfigure.SpringBootApplication;
4621    import org.springframework.boot.builder.SpringApplicationBuilder;
4622    import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
4623    import org.springframework.context.annotation.ComponentScan;
4624
4625    @SpringBootApplication
4626    @ComponentScan(basePackages = "com.example")
4627    public class SpringBootMembershipApplication extends SpringBootServletInitializer{
4628
4629        @Override
4630        protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
4631            return application.sources(SpringBootMembershipApplication.class);
4632        }
4633
4634        public static void main(String[] args) {
4635            SpringApplication.run(SpringBootMembershipApplication.class, args);
4636        }
4637    }
4638
4639 4)SpringBootMembership project > right-click > Maven > Update project > OK
4640
4641 5)pom.xml > right-click > Run As > Maven install
4642
4643    [INFO] BUILD SUCCESS
4644
4645 6)Tomcat에 이 Application을 Add 하여 test
4646 7)Tomcat Start
```

```

4647 8)SpringBootMembership project > right-click > Run As > Run on Server
4648 9)Finish
4649 10)http://localhost:8080/SpringBootMembership/
4650
4651
4652
4653 -----
4654 Task15. Spring Boot REST API 사용하기
4655 1. 위의 Task15에서 사용한 SpringBootMembership Application을 Copy후 Paste하여 이름변경
4656 2. SpringBootRestAPIDemo로 변경
4657 1)pom.xml
4658
4659     <groupId>com.example</groupId>
4660     <artifactId>SpringBootRestAPIDemo</artifactId>
4661     <version>0.0.1-SNAPSHOT</version>
4662     <packaging>war</packaging>
4663     <name>SpringBootRestAPIDemo</name>
4664     <description>Demo project for Spring Boot</description>
4665
4666 2)main method가 있는 Class의 이름 변경하기
4667
4668     package com.example.biz;
4669
4670     import org.springframework.boot.SpringApplication;
4671     import org.springframework.boot.autoconfigure.SpringBootApplication;
4672     import org.springframework.boot.builder.SpringApplicationBuilder;
4673     import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
4674     import org.springframework.context.annotation.ComponentScan;
4675
4676     @SpringBootApplication
4677     @ComponentScan(basePackages = "com.example")
4678     public class SpringBootRestAPIDemoApplication extends SpringBootServletInitializer{
4679
4680         @Override
4681         protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
4682             return application.sources(SpringBootRestAPIDemoApplication.class);
4683         }
4684
4685         public static void main(String[] args) {
4686             SpringApplication.run(SpringBootRestAPIDemoApplication.class, args);
4687         }
4688     }
4689
4690 3)server.xml 변경
4691
4692     <Context docBase="SpringBootRestAPIDemo" path="/SpringBootRestAPIDemo"
4693         reloadable="true" source="org.eclipse.jst.jee.server:SpringBootRestAPIDemo"/>
4694
4695 3. 전체 Member 조회하기
4696 1)MainController.java 수정하기
4697
4698     @Slf4j
4699     @RestController    <---변경
4700     public class MainController {
4701         @Autowired
4702         private MemberService memberService;

```

```

4704
4705 ...
4706 ...
4707 @RequestMapping(value = "/members", method = RequestMethod.GET)
4708 public Map mebers() {
4709     Map<String, Object> map = new HashMap<String, Object>();
4710     this.memberService.selectAllMembers(map);
4711     List<MemberVO> list = (List<MemberVO>)map.get("results");
4712     list.forEach(member -> log.info("'" + member));
4713     map.put("code", "success");
4714     return map;
4715 }
4716

```

2)POSTMAN에서 Test

GET <http://localhost:8080/SpringBootRestAPIDemo/members> Send

```

4719
4720 {
4721     "code": "success",
4722     "results": [
4723         {
4724             "userid": "developer",
4725             "passwd": "12345678",
4726             "name": "배지민",
4727             "age": 30,
4728             "gender": "0",
4729             "city": "대구"
4730         },
4731         {
4732             "userid": "jimin",
4733             "passwd": "12345678",
4734             "name": "한지민",
4735             "age": 24,
4736             "gender": "0",
4737             "city": "대전"
4738         }
4739     ]
4740 }
4741

```

3)src/main/resources/static > right-click > New > Other > Web > HTML File

-File name : index.html

```

4744
4745 <!DOCTYPE html>
4746 <html lang="en">
4747
4748 <head>
4749     <meta charset="utf-8">
4750     <meta name="viewport" content="width=device-width, initial-scale=1.0">
4751     <title>example.com 회원명단</title>
4752     <link rel="stylesheet" href="css/bootstrap.min.css">
4753     <link rel="stylesheet" href="css/bootstrap-theme.min.css">
4754     <style type="text/css">
4755         body {
4756             margin: 0px;
4757             padding: 0px;
4758         }
4759
4760         table {
4761             width: 500px;

```

```

4762         margin-left: auto;
4763         margin-right: auto;
4764         border-collapse: collapse;
4765         border: 1px solid white;
4766     }
4767
4768     th,
4769     td {
4770         border: 1px solid blue;
4771     }
4772
4773     th {
4774         background-color: navy;
4775         color: white;
4776         height: 40px;
4777         text-align: center;
4778     }
4779
4780     td {
4781         text-align: center;
4782     }
4783 </style>
4784 <script src="js/jquery-3.5.0.min.js"></script>
4785 <script>
4786     $(document)
4787         .ready(
4788             function () {
4789                 $
4790                     .ajax({
4791                         url: "/SpringBootRestAPIDemo/members",
4792                         type: "GET",
4793                         dataType: "json",
4794                         success: function (data) {
4795                             var str = "";
4796                             var members = data.results;
4797                             for (var i = 0; i < members.length; i++) {
4798                                 str += "<tr>";
4799                                 var userid = members[i].userid;
4800                                 var gender = null;
4801                                 if (members[i].gender == '1')
4802                                     gender = "남성";
4803                                 else
4804                                     gender = "여성";
4805                                 str += "<td><a href='display.html?userid=" + members[i].userid +
4806                                     ">" + members[i].userid + "</a></td>"
4807                                     + "<td>" + members[i].name + "</td>"
4808                                     + "<td>" + members[i].age + "</td>"
4809                                     + "<td>" + gender + "</td>"
4810                                     + "<td>" + members[i].city + "</td>";
4811                                 str += "</tr>";
4812                             }
4813                             $("#result").html(str);
4814                         });
4815                 });
4816 </script>
4817 </head>
4818

```

```

4819     <body>
4820         <div>
4821             <a href="/SpringBootRestAPIDemo"></a>
4822         </div>
4823         <h1 class="text-center">회원명단</h1>
4824         <div style="text-align: center">
4825             <a href="register.html">Member Add</a>
4826         </div>
4827         <table border="1">
4828             <thead>
4829                 <tr>
4830                     <th>아이디</th>
4831                     <th>이름</th>
4832                     <th>나이</th>
4833                     <th>성별</th>
4834                     <th>거주지</th>
4835                 </tr>
4836             </thead>
4837             <tbody id="result">
4838             </tbody>
4839         </table>
4840     </body>
4841
4842 </html>

```

4)MainController.java 수정 - 아래 코드 삭제

```

4846     @GetMapping("/")
4847     public String index() {
4848         return "index";
4849     }

```

4. 특정 Member 조회하기

1)MainController.java

```

4855     @RequestMapping(value = "/members/{userid}", method = RequestMethod.GET)
4856     public Map display1(@PathVariable String userid) {
4857         Map<String, Object> map = new HashMap<String, Object>(); map.put("userid",
            userid);
4858         this.memberService.selectMember(map);
4859         List<MemberVO> list = (List<MemberVO>)map.get("result");
4860         MemberVO member = list.get(0);
4861         log.info("user = {}", member);
4862         map.remove("userid");
4863         map.remove("result");
4864         map.put("result", member);
4865         map.put("code", "success");
4866         return map;
4867     }

```

2)POSTMAN에서 Test

```

4871     GET http://localhost:8080/SpringBootRestAPIDemo/members/jimin Send
4872
4873     {
4874         "result": [
4875             {

```



```

4876         "userid": "jimin",
4877         "passwd": "12345678",
4878         "name": "한지민",
4879         "age": 24,
4880         "gender": "0",
4881         "city": "대전"
4882     }
4883 ],
4884     "code": "success"
4885 }
4886
4887 3)static/display.html
4888
4889 <!DOCTYPE html>
4890 <html lang="en">
4891 <head>
4892     <meta charset="UTF-8">
4893     <meta name="viewport" content="width=device-width, initial-scale=1.0">
4894     <title>회원정보 상세보기</title>
4895     <script src="js/jquery-3.5.0.min.js"></script>
4896     <script>
4897         $(function () {
4898             userid = location.search.substring(1).split("=")[1];
4899             $.ajax({
4900                 url: "/SpringBootRestAPIDemo/members/" + userid,
4901                 type: "GET",
4902                 dataType : "json",
4903                 success: function (data) {
4904                     var member = data.result;
4905                     $(".userid").text(member.userid);
4906                     $(".name").text(member.name);
4907                     $(".#age").text(member.age);
4908                     if(member.gender == '1') $(".#gender").text("남성");
4909                     else if(member.gender == '0') $(".#gender").text("여성");
4910                     $(".#city").text(member.city);
4911                 },
4912                 error : function(err){
4913                     alert('error = ' + err);
4914                 }
4915             });
4916         });
4917     </script>
4918 </head>
4919
4920 <body>
4921     <div><a href="/SpringBootRestAPIDemo/"></a></div>
4923     <h1>Member Information</h1>
4924     <h1><span class="name"></span>(<span class="userid"></span>)님! 회원 정보</h1>
4925     <ul>
4926         <li>아이디 : <span class="userid"></span></li>
4927         <li>이름 : <span class="name"></span></li>
4928         <li>나이 : <span id="age"></span></li>
4929         <li>성별 : <span id="gender"></span></li>
4930         <li>거주지 : <span id="city"></span></li>
4931     </ul>
4932     <div>

```

```

4933         <a href="javascript:void(0)" onclick="javascript:history.back();" >목록으로</a>>
4934     </div>
4935 </body>
4936
4937 </html>
4938

```

4) <http://localhost:8080/SpringBootRestAPIDemo/display.html?userid=developer>

5. Member 등록 구현하기

1) MainController.java

```

4945 @RequestMapping(value = "/members", method = RequestMethod.POST)
4946 public Map register(@RequestBody MemberVO member) {
4947     log.info("member = {}", member);
4948     this.memberService.insertMember(member);
4949     Map<String, Object> map = new HashMap<String, Object>();
4950     map.put("code", "success");
4951     return map;
4952 }

```

2) Postman

POST <http://localhost:8080/SpringBootRestAPIDemo/members>

Body > raw > JSON(application/json)

```

{
  "userid" : "girlsage",
  "passwd" : "1234",
  "name" : "소녀시대",
  "age" : 30,
  "gender" : "0",
  "city" : "광주"
}

```

Send 버튼 클릭하면

```

Body
{"code": "success"}

```

3) static/register.html

```

4976 <!DOCTYPE html>
4977 <html lang="en">
4978 <head>
4979 <meta charset="UTF-8">
4980 <meta name="viewport" content="width=device-width, initial-scale=1.0">
4981 <title>회원가입</title>
4982 <link rel="stylesheet" href="css/bootstrap.min.css" >
4983 <script src="js/jquery-3.5.0.min.js"></script>
4984 <script>
4985     $(function() {
4986         var flag = 0;
4987         $("#userid").bind("change", function() {
4988             if ($("#userid").val()) {
4989                 $("#useriddiv").attr("class", "form-group has-success has-feedback");
4990                 $("#useridicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");

```

```

4991         $("#useridr").text("(success)");
4992         flag++;
4993     }
4994 });
4995 $("#passwd").bind("change", function() {
4996     if ($("#passwd").val()) {
4997         $("#passwddiv").attr("class", "form-group has-success has-feedback");
4998         $("#passwdicon").attr("class", "glyphicon glyphicon-ok
form-control-feedback");
4999         $("#passwdsr").text("(success)");
5000         flag++;
5001     }
5002 });
5003 $("#name").bind("change", function() {
5004     if ($("#name").val()) {
5005         $("#namediv").attr("class",
5006             "form-group has-success has-feedback");
5007         $("#nameicon")
5008             .attr("class",
5009                 "glyphicon glyphicon-ok form-control-feedback");
5010         $("#namesr").text("(success)");
5011         flag++;
5012     }
5013 });
5014 $("#age").bind("keyup", function() {
5015     if ($("#age").val() >= 20 && ($("#age").val() <= 65) {
5016         $("#agediv").attr("class", "form-group has-success has-feedback");
5017         $("#ageicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
5018         $("#agesr").text("(success)");
5019         flag++;
5020     }
5021 });
5022 $(":radio").bind("click", function() {
5023     $("#genderdiv").attr("class",
5024         "form-group has-success has-feedback");
5025     flag++;
5026 });
5027 $("#city").bind("change", function() {
5028     if ($("#city").val()) {
5029         $("#citydiv").attr("class",
5030             "form-group has-success has-feedback");
5031         flag++;
5032         if (flag == 6) {
5033             $('button').removeAttr("disabled");
5034         }
5035     }
5036 });
5037
5038 $("button[type='button']").bind("click", function(){
5039     $.ajax({
5040         url : "/SpringBootRestAPIDemo/members",
5041         contentType : "application/json;charset=utf-8",
5042         type : "POST",
5043         data : JSON.stringify({
5044             "userid" : $("#userid").val(),
5045             "passwd" : $("#passwd").val(),
5046             "name" : $("#name").val(),
5047             "age" : $("#age").val(),

```

```

5048         "gender" : $(".gender:checked").val(),
5049         "city" : $("#city").val()
5050     }},
5051     dataType : "json",
5052     success : function(data){
5053         alert(data.code);
5054         location.href = "/SpringBootRestAPIDemo/";
5055     }
5056     });
5057 });
5058 })
5059 </script>
5060 </head>
5061 <body>
5062     <div class="container">
5063         <div class="row">
5064             <h1 class="h1 text-center">회원가입</h1>
5065         </div>
5066         <div class="row">
5067             <form class="form-horizontal">
5068                 <div id="useriddiv" class="form-group has-error has-feedback">
5069                     <label for="userid" class="col-sm-3 control-label">User ID</label>
5070                     <div class="col-sm-6">
5071                         <div class="input-group">
5072                             <span class="input-group-addon">
5073                                 <span class="glyphicon glyphicon-trash"></span>
5074                             </span>
5075                             <input type="text" name="userid" id="userid" class="form-control"
5076                                 placeholder="Enter your ID">
5077                             <span id="useridicon" class="glyphicon glyphicon-remove
5078                                 form-control-feedback" aria-hidden="true"></span>
5079                             <span id="useridsr" class="sr-only">(error)</span>
5080                         </div>
5081                     <div id="passwddiv" class="form-group has-error has-feedback">
5082                         <label for="passwd" class="col-sm-3 control-label">Password</label>
5083                         <div class="col-sm-6">
5084                             <div class="input-group">
5085                                 <span class="input-group-addon">
5086                                     <span class="glyphicon glyphicon-lock"></span>
5087                                 </span>
5088                                 <input type="password" name="passwd" id="passwd"
5089                                     class="form-control" placeholder="Enter your password">
5090                                 <span id="passwdicon" class="glyphicon glyphicon-remove
5091                                     form-control-feedback" aria-hidden="true"></span>
5092                                 <span id="passwdsr" class="sr-only">(error)</span>
5093                             </div>
5094                         <div id="namediv" class="form-group has-error has-feedback">
5095                             <label for="name" class="col-sm-3 control-label">Name</label>
5096                             <div class="col-sm-6">
5097                                 <div class="input-group">
5098                                     <span class="input-group-addon">
5099                                         <span class="glyphicon glyphicon-user"></span>
5100                                     </span>
5101                                     <input type="text" name="name" id="name" class="form-control"

```

```

placeholder="Enter your name">
5102     </div>
5103     <span id="nameicon" class="glyphicon glyphicon-remove
form-control-feedback" aria-hidden="true"></span>
5104     <span id="namesr" class="sr-only">(error)</span>
5105     </div>
5106 </div>
5107 <div id="agediv" class="form-group has-error has-feedback">
5108     <label for="age" class="col-sm-3 control-label">Age</label>
5109     <div class="col-sm-4">
5110         <div class="input-group">
5111             <span class="input-group-addon">
5112                 <span class="glyphicon glyphicon-music"></span>
5113             </span>
5114             <input type="number" name="age" id="age" class="form-control"
placeholder="Enter your age">
5115         </div>
5116         <span id="ageicon" class="glyphicon glyphicon-remove
form-control-feedback" aria-hidden="true"></span>
5117         <span id="agesr" class="sr-only">(error)</span>
5118     </div>
5119 </div>
5120 <div id="genderdiv" class="form-group has-error has-feedback">
5121     <label class="col-sm-3 control-label">Gender</label>
5122     <div class="col-sm-6">
5123         <label class="radio-inline">
5124             <input type="radio" name="gender" value="1" class="gender">Male
5125         </label>
5126         <label class="radio-inline">
5127             <input type="radio" name="gender" value="0" class="gender">Female
5128         </label>
5129     </div>
5130 </div>
5131 <div id="citydiv" class="form-group has-error has-feedback">
5132     <label for="city" class="col-sm-3 control-label">City</label>
5133     <div class="col-sm-6">
5134         <select name="city" id="city" class="form-control">
5135             <option value="">--선택--</option>
5136             <option value="서울">서울</option>
5137             <option value="부산">부산</option>
5138             <option value="대전">대전</option>
5139             <option value="광주">광주</option>
5140             <option value="대구">대구</option>
5141         </select>
5142     </div>
5143 </div>
5144 <div class="form-group">
5145     <div class="col-sm-6 col-sm-offset-3">
5146         <button type="button" class="col-sm-4 btn btn-primary"
disabled>Submit</button>
5147     </div>
5148 </div>
5149 </form>
5150 </div>
5151 </div>
5152 </body>
5153 </html>
5154

```

[illegible]

```

5212     });
5213     });
5214     </script>
5215
5216 4)templates/update.html
5217
5218 <!DOCTYPE html>
5219 <html lang="en">
5220 <head>
5221     <meta charset="UTF-8">
5222     <meta name="viewport" content="width=device-width, initial-scale=1.0">
5223     <title>회원정보 수정창</title>
5224     <link rel="stylesheet" href="css/bootstrap.min.css">
5225     <script src="js/jquery-3.5.0.min.js"></script>
5226     <script>
5227         $(function () {
5228             userid = location.search.substring(1).split("=")[1];
5229             $.ajax({
5230                 url: "/SpringBootRestAPIDemo/members/" + userid,
5231                 type: "GET",
5232                 dataType : "json",
5233                 success: function (data) {
5234                     var member = data.result;
5235                     $("#userid").val(member.userid);
5236                     $("#name").val(member.name);
5237                     $("#age").val(member.age);
5238                     if(member.gender == '1') $("#male").prop("checked", true);
5239                     else if(member.gender == '0') $("#female").prop("checked", true);
5240                     switch(member.city){
5241                         case "서울" : $("#city option:eq(1)").attr("selected", "selected"); break;
5242                         case "부산" : $("#city option:eq(2)").attr("selected", "selected"); break;
5243                         case "대전" : $("#city option:eq(3)").attr("selected", "selected"); break;
5244                         case "광주" : $("#city option:eq(4)").attr("selected", "selected"); break;
5245                         case "대구" : $("#city option:eq(5)").attr("selected", "selected"); break;
5246                     }
5247                 },
5248                 error : function(err){
5249                     alert('error = ' + err);
5250                 }
5251             });
5252
5253             $("#useriddiv").attr("class", "form-group has-success has-feedback");
5254             $("#useridicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
5255             $("#useridsr").text("(success)");
5256
5257             $("#passwddiv").attr("class", "form-group has-success has-feedback");
5258             $("#passwdicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
5259             $("#passwdsr").text("(success)");
5260
5261             $("#namediv").attr("class", "form-group has-success has-feedback");
5262             $("#nameicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
5263             $("#namesr").text("(success)");
5264
5265             $("#agediv").attr("class", "form-group has-success has-feedback");
5266             $("#ageicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
5267             $("#agesr").text("(success)");
5268
5269             $("#genderdiv").attr("class", "form-group has-success has-feedback");

```

```

5270
5271     $("#citydiv").attr("class", "form-group has-success has-feedback");
5272
5273     $("#button[type=button]").bind("click", function(){
5274         $.ajax({
5275             url : "/SpringBootRestAPIDemo/members",
5276             type : "PUT",
5277             data : JSON.stringify({
5278                 "age" : $("#age").val(),
5279                 "gender" : $(".gender:checked").val(),
5280                 "city" : $("#city").val(),
5281                 "userid" : userid
5282             }),
5283             contentType : "application/json;charset=utf-8",
5284             success : function(data){
5285                 alert(data.code);
5286                 location.href = "/SpringBootRestAPIDemo/";
5287             }
5288         });
5289     });
5290 })
5291 </script>
5292 </head>
5293 <body>
5294     <div class="container">
5295         <div class="row">
5296             <h1 class="h1 text-center">회원 정보 수정</h1>
5297         </div>
5298         <div class="row">
5299             <form class="form-horizontal">
5300                 <div id="useriddiv" class="form-group has-error has-feedback">
5301                     <label for="userid" class="col-sm-3 control-label">User ID</label>
5302                     <div class="col-sm-6">
5303                         <div class="input-group">
5304                             <span class="input-group-addon">
5305                                 <span class="glyphicon glyphicon-trash"></span>
5306                             </span>
5307                             <input type="text" name="userid" id="userid" class="form-control"
5308                                 placeholder="Enter your ID">
5309                             </div>
5310                             <span id="useridicon" class="glyphicon glyphicon-remove
5311                                 form-control-feedback" aria-hidden="true"></span>
5312                             <span id="useridsr" class="sr-only">(error)</span>
5313                         </div>
5314                     </div>
5315                 <div id="passwddiv" class="form-group has-error has-feedback">
5316                     <label for="passwd" class="col-sm-3 control-label">Password</label>
5317                     <div class="col-sm-6">
5318                         <div class="input-group">
5319                             <span class="input-group-addon">
5320                                 <span class="glyphicon glyphicon-lock"></span>
5321                             </span>
5322                             <input type="password" name="passwd" id="passwd"
5323                                 class="form-control" placeholder="Enter your password">
5324                             </div>
5325                             <span id="passwdicon" class="glyphicon glyphicon-remove
5326                                 form-control-feedback" aria-hidden="true"></span>
5327                             <span id="passwdsr" class="sr-only">(error)</span>
5328                         </div>
5329                     </div>
5330                 </form>
5331             </div>

```



```
5324         </div>
5325     </div>
5326     <div id="namediv" class="form-group has-error has-feedback">
5327         <label for="name" class="col-sm-3 control-label">Name</label>
5328         <div class="col-sm-6">
5329             <div class="input-group">
5330                 <span class="input-group-addon">
5331                     <span class="glyphicon glyphicon-user"></span>
5332                 </span>
5333                 <input type="text" name="name" id="name" class="form-control"
                    placeholder="Enter your name">
5334             </div>
5335             <span id="nameicon" class="glyphicon glyphicon-remove
                    form-control-feedback" aria-hidden="true"></span>
5336             <span id="namesr" class="sr-only">(error)</span>
5337         </div>
5338     </div>
5339     <div id="agediv" class="form-group has-error has-feedback">
5340         <label for="age" class="col-sm-3 control-label">Age</label>
5341         <div class="col-sm-4">
5342             <div class="input-group">
5343                 <span class="input-group-addon">
5344                     <span class="glyphicon glyphicon-music"></span>
5345                 </span>
5346                 <input type="number" name="age" id="age" class="form-control"
                    placeholder="Enter your age">
5347             </div>
5348             <span id="ageicon" class="glyphicon glyphicon-remove
                    form-control-feedback" aria-hidden="true"></span>
5349             <span id="agesr" class="sr-only">(error)</span>
5350         </div>
5351     </div>
5352     <div id="genderdiv" class="form-group has-error has-feedback">
5353         <label class="col-sm-3 control-label">Gender</label>
5354         <div class="col-sm-6">
5355             <label class="radio-inline">
5356                 <input type="radio" name="gender" value="1" id="male"
                    class="gender">Male
5357             </label>
5358             <label class="radio-inline">
5359                 <input type="radio" name="gender" value="0" id="female"
                    class="gender">Female
5360             </label>
5361         </div>
5362     </div>
5363     <div id="citydiv" class="form-group has-error has-feedback">
5364         <label for="city" class="col-sm-3 control-label">City</label>
5365         <div class="col-sm-6">
5366             <select name="city" id="city" class="form-control">
5367                 <option value="">--선택--</option>
5368                 <option value="서울">서울</option>
5369                 <option value="부산">부산</option>
5370                 <option value="대전">대전</option>
5371                 <option value="광주">광주</option>
5372                 <option value="대구">대구</option>
5373             </select>
5374         </div>
5375     </div>
```

5386

5387

5389 1)MainController.java

5390

5398

5400

5405

5407 -아래의 code를 추가한다.

5411

5412

5413

```
5415         if(confirm("정말 탈퇴하시겠습니까?")){
```

5423

```
5425         location.go(0);
```

5427 });

5428

5429

5430

5431

5152

```
5433 1. 위 Task15 Project를 다음과 같이 4개의 tier로 구성한다.
5434 1)Client(Windows, Google Chrome Browser, 192.168.56.1)
5435 2)Web Server(Ubuntu Server, Apache2, /WebHome/SpringBootRestAPIDemo/,
5436 192.168.56.2)
5437 3)WAS(Ubuntu Server, Tomcat9, /etc/tomcat9/webapps/SpringBootRestAPIDemo.war,
5438 192.168.56.3)
5439 4)DB(CentOS 7, Oracle 18C XE, 192.168.56.4)
5440 2. Web Server
5441 -src/main/resoruces/static folder의 다음 resource를 /WebHome/SpringBootRestAPIDemo/으로
5442 복사한다.
5443 1)css
5444 -bootstrap.min.cs
5445 -bootstrap-theme.min.css
5446 2)fonts
5447 -glyphicons-halflings-regular.eot
5448 -glyphicons-halflings-regular.svg
5449 -glyphicons-halflings-regular.ttf
5450 -glyphicons-halflings-regular.woff
5451 -glyphicons-halflings-regular.woff2
5452
5453 3)images
5454 -spring-boot.png
5455
5456 4)js
5457 -bootstrap.min.js
5458 -jquery-3.5.0.min.js
5459
5460 5)display.html
5461
5462 <script>
5463 $(function () {
5464     userid = location.search.substring(1).split("=")[1];
5465     $.ajax({
5466         url: "http://192.168.56.2:8080/SpringBootRestAPIDemo/members/" + userid,
5467         type: "GET",
5468         dataType : "json",
5469         success: function (data) {
5470             var member = data.result;
5471             $(".userid").text(member.userid);
5472             $(".name").text(member.name);
5473             $(".#age").text(member.age);
5474             if(member.gender == '1') $(".#gender").text("남성");
5475             else if(member.gender == '0') $(".#gender").text("여성");
5476             $(".#city").text(member.city);
5477         },
5478         error : function(err){
5479             alert('error = ' + err);
5480         }
5481     });
5482     $(".#btnUpdate").bind("click", function(){
5483         location.href = "update.html?userid=" + userid;
5484     });
5485
5486     $(".#btnDelete").bind("click", function(){
5487         if(confirm("정말 탈퇴하시겠습니까?")){
```

```

5488     $.ajax({
5489         url : "http://192.168.56.2:8080/SpringBootRestAPIDemo/members/" + userid,
5490         type : "DELETE",
5491         success : function(data){
5492             alert(data.code);
5493             location.href = "/SpringBootRestAPIDemo/";
5494         }
5495     });
5496 }else{
5497     location.go(0);
5498 }
5499 });
5500 });
5501 </script>

```

6)index.html

```

5505 <script>
5506     $(document)
5507     .ready(
5508         function () {
5509             $
5510             .ajax({
5511                 url: "http://192.168.56.2:8080/SpringBootRestAPIDemo/members",
5512                 type: "GET",
5513                 dataType: "json",
5514                 success: function (data) {
5515                     var str = "";
5516                     var members = data.results;
5517                     for (var i = 0; i < members.length; i++) {
5518                         str += "<tr>";
5519                         var userid = members[i].userid;
5520                         var gender = null;
5521                         if (members[i].gender == '1')
5522                             gender = "남성";
5523                         else
5524                             gender = "여성";
5525                         str += "<td><a href='display.html?userid=" + members[i].userid + ">"
5526                             + members[i].userid + "</a></td>"
5527                             + "<td>" + members[i].name + "</td>"
5528                             + "<td>" + members[i].age + "</td>"
5529                             + "<td>" + gender + "</td>"
5530                             + "<td>" + members[i].city + "</td>";
5531                         str += "</tr>";
5532                     }
5533                     $("#result").html(str);
5534                 }
5535             });
5536         }
5537     </script>

```

7)register.html

```

5539 <script>
5540     ...
5541     $("button[type='button']").bind("click", function(){
5542         $.ajax({
5543             url : "http://192.168.56.2:8080/SpringBootRestAPIDemo/members",

```

```

5545     contentType : "application/json;charset=utf-8",
5546     type : "POST",
5547     data : JSON.stringify({
5548         "userid" : $("#userid").val(),
5549         "passwd" : $("#passwd").val(),
5550         "name" : $("#name").val(),
5551         "age" : $("#age").val(),
5552         "gender" : ($("#gender:checked").val(),
5553         "city" : ($("#city").val()
5554     })),
5555     dataType : "json",
5556     success : function(data) {
5557         alert(data.code);
5558         location.href = "/SpringBootRestAPIDemo/";
5559     }
5560     });
5561 });
5562 })
5563 </script>
5564
5565 8)update.html
5566
5567 <script>
5568     $(function () {
5569         userid = location.search.substring(1).split("=")[1];
5570         $.ajax({
5571             url: "http://192.168.56.2:8080/SpringBootRestAPIDemo/members/" + userid,
5572             type: "GET",
5573             dataType : "json",
5574             success: function (data) {
5575                 var member = data.result;
5576                 $("#userid").val(member.userid);
5577                 $("#name").val(member.name);
5578                 $("#age").val(member.age);
5579                 if(member.gender == '1') $("#male").prop("checked", true);
5580                 else if(member.gender == '0') $("#female").prop("checked", true);
5581                 switch(member.city){
5582                     case "서울" : $("#city option:eq(1)").attr("selected", "selected"); break;
5583                     case "부산" : $("#city option:eq(2)").attr("selected", "selected"); break;
5584                     case "대전" : $("#city option:eq(3)").attr("selected", "selected"); break;
5585                     case "광주" : $("#city option:eq(4)").attr("selected", "selected"); break;
5586                     case "대구" : $("#city option:eq(5)").attr("selected", "selected"); break;
5587                 }
5588             },
5589             error : function(err){
5590                 alert('error = ' + err);
5591             }
5592         });
5593
5594         $("#useriddiv").attr("class", "form-group has-success has-feedback");
5595         $("#useridicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
5596         $("#useridsr").text("(success)");
5597
5598         $("#passwddiv").attr("class", "form-group has-success has-feedback");
5599         $("#passwdicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
5600         $("#passwdsr").text("(success)");
5601
5602         $("#namediv").attr("class", "form-group has-success has-feedback");

```

```

5603     $("#nameicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
5604     $("#namesr").text("(success)");
5605
5606     $("#agediv").attr("class", "form-group has-success has-feedback");
5607     $("#ageicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
5608     $("#agesr").text("(success)");
5609
5610     $("#genderdiv").attr("class", "form-group has-success has-feedback");
5611
5612     $("#citydiv").attr("class", "form-group has-success has-feedback");
5613
5614     $("#button[type=button]").bind("click", function(){
5615         $.ajax({
5616             url : "http://192.168.56.2:8080/SpringBootRestAPIDemo/members",
5617             type : "PUT",
5618             data : JSON.stringify({
5619                 "age" : $("#age").val(),
5620                 "gender" : $(".gender:checked").val(),
5621                 "city" : $("#city").val(),
5622                 "userid" : userid
5623             }),
5624             contentType : "application/json;charset=utf-8",
5625             success : function(data){
5626                 alert(data.code);
5627                 location.href = "/SpringBootRestAPIDemo/";
5628             }
5629         });
5630     });
5631 }
5632 </script>
5633
5634
5635

```

3. WAS

- 5636 1)CORS는 동일한 출처(Origin: 최초 자원이 서비스된 출처)가 아니어도 다른 출처에서의 자원을 요청하여 쓸 수 있게 허용하는 구조를 뜻한다.
- 5637 2)보통 보안 상의 이슈(DOM을 통한 취약한 데이터 접근 시도) 때문에 동일 출처(Single Origin Policy)를 기본적으로 웹에서는 준수한다.
- 5638 3)따라서 최초 자원을 요청한 출처 말고 다른 곳으로 Script를 통해 자원을 요청하는 것은 금지된다.
- 5639 4)CORS를 적용하려면 Web Application에 그에 따른 처리를 해야하고 Spring Boot에서는 @CrossOrigin 혹은 WebConfig를 통해 CORS를 적용하는 방법을 제공한다.

5640

5641 5)@CrossOrigin annotation

```

5642
5643 -MainController.java
5644
5645 @Slf4j
5646 @CrossOrigin(origins="http://localhost") <---자원을 요청한 쪽의 URI
5647 @RestController
5648 public class MainController {
5649     @Autowired
5650     private MemberService memberService;
5651
5652

```

5653 6)WebConfig

```

5654
5655 -MainController.java
5656
5657 @Slf4j

```

```

5658     @RestController
5659     public class MainController {
5660
5661         @Autowired
5662         private MemberService memberService;
5663
5664     -WebMvcConfigurer.java
5665
5666     @Configuration
5667     public class WebConfig implements WebMvcConfigurer {
5668
5669         @Override
5670         public void addCorsMappings(CorsRegistry registry) {
5671             registry.addMapping("/**")
5672                 .allowedOrigins("http://localhost");
5673         }
5674     }
5675
5676
5677

```

5678 -----

5679 Task17. Spring Boot와 JPA 연동하기

- 5680 1. Spring Boot의 Database 처리는 기본적으로 JPA 기술을 기반으로 하고 있다.
- 5681 2. 이 JPA를 Spring Framework에서 사용할 수 있게 한 것이 'Spring Data JPA' framework이다.
- 5682 3. Spring Boot에서는 JTA(Java Transaction API; Java EE에 transaction 처리를 제공), Spring ORM, Spring Aspects/Spring AOP는 'Spring Boot Starter Data JPA'라는 library를 사용해서 통합적으로 사용할 수 있다.
- 5683 4. 즉, 이 library는 각종 library를 조합해서 간단히 database 접속을 구현하게 한 기능이다.

5684 5. Spring Boot project 생성

5685 1)Package Explorer > right-click > New > Spring Starter Project

5686 2)다음 각 항목의 값을 입력한 후 Next 클릭

5687 -Service URL :<http://start.spring.io>

5688 -Name : JpaDemo

5689 -Type : Maven

5690 -Packaging : Jar

5691 -Java Version : 8

5692 -Language : Java

5693 -Group : com.example

5694 -Artifact : JpaDemo

5695 -Version : 0.0.1-SNAPSHOT

5696 -Description : Demo project for Spring Boot

5697 -Package : com.example.biz

5698

5699 3)다음 각 항목을 선택한 후 Finish 클릭

5700 -Spring Boot Version : 2.2.4

5701 -Select

5702 --SQL > Spring Data JPA, H2 Database

5703 --Developer Tools > Spring Boot DevTools

5704 -Finish

5705

5706

5707

5708 6. Entity

5709 1)JPA에서 Entity라는 것은 Database에 저장하기 위해서 정의한 class이다.

5710 2)일반적으로 RDBMS에서 Table 같은 것이다.

5711 3)com.example.biz > right-click > New > Class

5712

5713 4)Name : MemberVO

```
5714
5715     package com.example.biz;
5716
5717     import javax.persistence.Column;
5718     import javax.persistence.Entity;
5719     import javax.persistence.GeneratedValue;
5720     import javax.persistence.GenerationType;
5721     import javax.persistence.Id;
5722
5723
5724     @Entity(name="Member")
5725     public class MemberVO {
5726         @Id
5727         @GeneratedValue(strategy = GenerationType.AUTO)
5728         private long id;
5729         @Column
5730         private String username;
5731         @Column
5732         private int age;
5733
5734         public MemberVO() {}
5735
5736         public MemberVO(String username, int age) {
5737             this.username = username;
5738             this.age = age;
5739         }
5740
5741         public long getId() {
5742             return id;
5743         }
5744
5745         public void setId(long id) {
5746             this.id = id;
5747         }
5748
5749         public String getUsername() {
5750             return username;
5751         }
5752
5753         public void setUsername(String username) {
5754             this.username = username;
5755         }
5756
5757         public int getAge() {
5758             return age;
5759         }
5760
5761         public void setAge(int age) {
5762             this.age = age;
5763         }
5764
5765         @Override
5766         public String toString() {
5767             return "MemberVO [id=" + id + ", username=" + username + ", age=" + age + "];"
5768         }
5769     }
5770
5771     5)여기서 주의할 점은 기본 생성자는 반드시 넣어야 한다.
```


5772

5773

5774 7. Repository

5775 1)Entity class를 구성했다면 이번엔 Repository interface를 만들어야 한다.

5776 2)Spring Framework에서는 Entity의 기본적인 삽입, 조회, 수정, 삭제가 가능하도록 CrudRepository라는 interface가 있다.

5777 3)com.example.biz > right-click > New > Interface

5778

5779 4)Name : MemberRepository

5780

5781 package com.example.biz;

5782

5783 import java.util.List;

5784

5785 import org.springframework.data.jpa.repository.Query;

5786 import org.springframework.data.repository.CrudRepository;

5787 import org.springframework.data.repository.query.Param;

5788

5789 public interface MemberRepository extends CrudRepository<MemberVO, Long> {

5790 List<MemberVO> findByUsernameAndAgeLessThan(String username, int age);

5791

5792 @Query("select t from Member t where username= :username and age < :age")

5793 List<MemberVO> findByUsernameAndAgeLessThanSQL(@Param("username") String
username, @Param("age") int age);

5794

5795 List<MemberVO> findByUsernameAndAgeLessThanOrderByAgeDesc(String username,
int age);

5796 }

5797

5798 -위의 코드는 실제로 MemberVO Entity를 이용하기 위한 Repository class이다.

5799 -기본적인 method 외에도 추가적인 method를 지정할 수 있다.

5800 -method 이름을 기반(Query Method)으로 해서 만들어도 되고 @Query를 이용해 기존의 SQL처럼 만들어도 된다.

5801 -findByUsernameAndAgeLessThan method와 findByUsernameAndAgeLessThanSQL method
는 같은 결과를 출력하지만 전자의 method는 method 이름을 기반으로 한 것이고 후자의 method는 @Query
annotation을 기반으로 해서 만든 것이다.5802 -method 이름 기반으로 해서 만들면 추후에 사용할 때 method 이름만으로도 어떤 query인지 알 수 있다는 장
점이 있다.5803 -반대로 @Query annotation으로 만든 method는 기존의 source를 converting하는 경우 유용하게 사용할
수 있다.5804 -@Query5805 --다만 @Query annotation으로 query를 만들 때에 from 절에 들어가는 table은 Entity로 지정된 class
이름이다.

5806 -method 이름 기반 작성법

5807 -해당 부분은 Spring 문서

([https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.
query-creation](https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.query-creation))를 통해 확인할 수 있다.

5808

5809

5810 8. Application 작성

5811 1)Entity 및 Repository가 준비 되었다면 실제로 @SpringBootApplication annotation이 있는 class에서
실제로 사용해 보자.

5812

5813 package com.example.biz;

5814

5815 import java.util.List;

5816

5817 import org.springframework.beans.factory.annotation.Autowired;

```

5818 import org.springframework.boot.CommandLineRunner;
5819 import org.springframework.boot.SpringApplication;
5820 import org.springframework.boot.autoconfigure.SpringBootApplication;
5821
5822 @SpringBootApplication
5823 public class JpaDemoApplication implements CommandLineRunner {
5824     @Autowired
5825     MemberRepository memberRepository;
5826
5827     public static void main(String[] args) {
5828         SpringApplication.run(JpaDemoApplication.class, args);
5829     }
5830
5831     @Override
5832     public void run(String... args) throws Exception {
5833         memberRepository.save(new MemberVO("a", 10));
5834         memberRepository.save(new MemberVO("b", 15));
5835         memberRepository.save(new MemberVO("c", 10));
5836         memberRepository.save(new MemberVO("a", 5));
5837         Iterable<MemberVO> list1 = memberRepository.findAll();
5838         System.out.println("findAll() Method.");
5839         for (MemberVO m : list1) {
5840             System.out.println(m.toString());
5841         }
5842         System.out.println("findByUserNameAndAgeLessThan() Method.");
5843         List<MemberVO> list2 = memberRepository.findByUsernameAndAgeLessThan("a",
5844             10);
5845         for (MemberVO m : list2) {
5846             System.out.println(m.toString());
5847         }
5848         System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
5849         List<MemberVO> list3 =
5850             memberRepository.findByUsernameAndAgeLessThanSQL("a", 10);
5851         for (MemberVO m : list3) {
5852             System.out.println(m.toString());
5853         }
5854         System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
5855         List<MemberVO> list4 =
5856             memberRepository.findByUsernameAndAgeLessThanOrderByAgeDesc("a", 15);
5857         for (MemberVO m : list4) {
5858             System.out.println(m.toString());
5859         }
5860         memberRepository.deleteAll();
5861     }
5862 }
5863

```

9. 실행

```

5866 findAll() Method.
5867 MemberVO [id=1, username=a, age=10]
5868 MemberVO [id=2, username=b, age=15]
5869 MemberVO [id=3, username=c, age=10]
5870 MemberVO [id=4, username=a, age=5]
5871 findByUserNameAndAgeLessThan() Method.
5872 MemberVO [id=4, username=a, age=5]

```

```
5873 findByUsernameAndAgeLessThanSQL() Method.
5874 MemberVO [id=4, username=a, age=5]
5875 findByUsernameAndAgeLessThanSQL() Method.
5876 MemberVO [id=1, username=a, age=10]
5877 MemberVO [id=4, username=a, age=5]
5878
5879
5880
5881 -----
5882 Task18. Spring Boot JPA
5883 1. Spring Boot project 생성
5884 1)Package Explorer > right-click > New > Spring Starter Project
5885
5886 2)다음의 각 항목의 값을 입력 후 Next 클릭
5887 -Service URL :http://start.spring.io
5888 -Name : BootJpaDemo
5889 -Type : Maven
5890 -Packaging : Jar
5891 -Java Version : 8
5892 -Language : Java
5893 -Group : com.example
5894 -Artifact : BootJpaDemo
5895 -Version : 0.0.1-SNAPSHOT
5896 -Description : Demo project for Spring Boot
5897 -Package : com.example.biz
5898
5899 3)각 항목을 선택 후 Finish 클릭
5900 -Spring Boot Version : 2.2.4
5901 -Select
5902 --SQL > Spring Data JPA, H2 Database
5903 --Developer Tools > Spring Boot DevTools
5904 --Web > Spring Web
5905 --Template Engines > Thymeleaf
5906 -Finish
5907
5908 4)DevTools
5909 -It provides developer tools.
5910 -These tools are helpful in application development mode.
5911 -One of the features of developer tool is automatic restart of the server for any change
    in code
5912
5913
5914 2. Entity 작성하기
5915 1)com.example.biz > right-click > New > Package
5916 2)Name : com.example.biz.vo
5917 3)com.example.biz.vo > right-click > New > Class
5918 4)Name : User
5919
5920 package com.example.biz.vo;
5921
5922 import javax.persistence.Column;
5923 import javax.persistence.Entity;
5924 import javax.persistence.GeneratedValue;
5925 import javax.persistence.GenerationType;
5926 import javax.persistence.Id;
5927 import javax.persistence.Table;
5928
5929 import lombok.Data;
```

```
5930
5931 @Entity
5932 @Table
5933 public class User {
5934
5935     @Id
5936     @GeneratedValue(strategy = GenerationType.AUTO)
5937     @Column
5938     private long id;
5939
5940     @Column(length = 20, nullable = false)
5941     private String username;
5942
5943     @Column(length = 100, nullable = true)
5944     private String email;
5945
5946     @Column(nullable = true)
5947     private Integer age;
5948
5949     @Column(nullable = true)
5950     private String memo;
5951
5952     public long getId() {
5953         return id;
5954     }
5955
5956     public void setId(long id) {
5957         this.id = id;
5958     }
5959
5960     public String getUsername() {
5961         return username;
5962     }
5963
5964     public void setUsername(String username) {
5965         this.username = username;
5966     }
5967
5968     public String getEmail() {
5969         return email;
5970     }
5971
5972     public void setEmail(String email) {
5973         this.email = email;
5974     }
5975
5976     public Integer getAge() {
5977         return age;
5978     }
5979
5980     public void setAge(Integer age) {
5981         this.age = age;
5982     }
5983
5984     public String getMemo() {
5985         return memo;
5986     }
5987 }
```

```
5988     public void setMemo(String memo) {
5989         this.memo = memo;
5990     }
5991 }
5992
5993
5994 3. Repository 생성하기
5995 1)com.example.biz > right-click > New > Package
5996 2)Name : com.example.biz.dao
5997 3)com.example.biz.dao > right-click > New > Interface
5998 4)Name : UserRepository > Finish
5999
6000     package com.example.biz.dao;
6001
6002     import org.springframework.data.jpa.repository.JpaRepository;
6003     import org.springframework.stereotype.Repository;
6004
6005     import com.example.biz.vo.User;
6006
6007     @Repository("repository")
6008     public interface UserRepository extends JpaRepository<User, Long>{
6009
6010     }
6011
6012     -JpaRepository라는 interface는 새로운 repository를 생성하기 위한 토대가 된다.
6013     -모든 Repository는 이 JpaRepository를 상속해서 작성한다
6014
6015
6016 4. HelloController 작성
6017 1)com.example.biz > right-click > New > Class
6018 2)Name : HelloController
6019
6020     package com.example.biz;
6021
6022     import org.springframework.beans.factory.annotation.Autowired;
6023     import org.springframework.stereotype.Controller;
6024     import org.springframework.web.bind.annotation.RequestMapping;
6025     import org.springframework.web.servlet.ModelAndView;
6026
6027     import com.example.biz.dao.UserRepository;
6028     import com.example.biz.vo.User;
6029
6030     @Controller
6031     public class HelloController {
6032
6033         @Autowired
6034         UserRepository repository;
6035
6036         @RequestMapping("/")
6037         public ModelAndView index(ModelAndView mav) {
6038             mav.setViewName("index");
6039             mav.addObject("msg", "this is sample content.");
6040             Iterable<User> list = repository.findAll();
6041             mav.addObject("data", list);
6042             return mav;
6043         }
6044     }
6045
```

6046 -UserRepository에는 findAll 같은 method가 정의되어 있지 않다.
6047 -이것은 부모 interface인 JpaRepository가 가지고 있는 method이다.
6048 -이를 통해 모든 entity가 자동으로 추출되는 것이다.
6049
6050
6051 5. JUnit Test
6052 1)src/test/java/com.example.biz.BootJpaDemoApplicationTests.java > right-click > Run As
6053 > JUnit Test
6054 2)Green bar
6055
6056 6. template 준비하기
6057 1)src/main/resources > right-click > New > File
6058
6059 2)File name : messages.properties
6060 content.title=Message sample page.
6061 content.message=This is sample message from properties.
6062
6063 3)src/main/resources/templates > right-click > New > Web > HTML Files
6064 4)Name : index.html
6065
6066 <!DOCTYPE HTML>
6067 <html xmlns:th="http://www.thymeleaf.org">
6068 <head>
6069 <title>top page</title>
6070 <meta charset="UTF-8" />
6071 <style>
6072 h1 { font-size:18pt; font-weight:bold; color:gray; }
6073 body { font-size:13pt; color:gray; margin:5px 25px; }
6074 pre { border: solid 3px #ddd; padding: 10px; }
6075 </style>
6076 </head>
6077 <body>
6078 <h1 th:text="#{content.title}">Hello page</h1>
6079 <pre th:text="\${data}"></pre>
6080 </body>
6081 </html>
6082
6083
6084 7. 실행해서 접속
6085 1)저장돼있는 data가 회색 사각 틀 안에 표시된다.
6086 2)아직 아무 data가 없기 때문에 빈 배열 []라고 표시된다.
6087
6088
6089 8. Entity의 CRUD 처리하기 : form으로 data 저장하기
6090 1)index.html 수정
6091
6092 <!DOCTYPE HTML>
6093 <html xmlns:th="http://www.thymeleaf.org">
6094 <head>
6095 <title>top page</title>
6096 <meta charset="UTF-8" />
6097 <style>
6098 h1 {
6099 font-size: 18pt;
6100 font-weight: bold;
6101 color: gray;
6102 }

```

6103
6104     body {
6105         font-size: 13pt;
6106         color: gray;
6107         margin: 5px 25px;
6108     }
6109
6110     tr {
6111         margin: 5px;
6112     }
6113
6114     th {
6115         padding: 5px;
6116         color: white;
6117         background: darkgray;
6118     }
6119
6120     td {
6121         padding: 5px;
6122         color: black;
6123         background: #e0e0ff;
6124     }
6125 </style>
6126 </head>
6127 <body>
6128     <h1 th:text="#{content.title}">Hello page</h1>
6129     <table>
6130         <form method="post" action="/" th:object="${formModel}">
6131             <tr>
6132                 <td><label for="username">이름</label></td>
6133                 <td><input type="text" name="username" th:value="*{username}" /></td>
6134             </tr>
6135             <tr>
6136                 <td><label for="age">연령</label></td>
6137                 <td><input type="text" name="age" th:value="*{age}" /></td>
6138             </tr>
6139             <tr>
6140                 <td><label for="email">메일</label></td>
6141                 <td><input type="text" name="email" th:value="*{email}" /></td>
6142             </tr>
6143             <tr>
6144                 <td><label for="memo">메모</label></td>
6145                 <td><textarea name="memo" th:text="*{memo}" cols="20"
6146                     rows="5"></textarea></td>
6147             </tr>
6148             <td></td>
6149             <td><input type="submit" /></td>
6150         </form>
6151     </table>
6152     <hr />
6153     <table>
6154         <tr>
6155             <th>ID</th>
6156             <th>이름</th>
6157         </tr>
6158         <tr th:each="obj : ${datalist}">

```

```
6160         <td th:text="${obj.id}"></td>
6161         <td th:text="${obj.username}"></td>
6162     </tr>
6163 </table>
6164 </body>
6165 </html>
6166
6167 2)HelloController.java 수정
6168
6169     package com.example.biz;
6170
6171     import org.springframework.beans.factory.annotation.Autowired;
6172     import org.springframework.stereotype.Controller;
6173     import org.springframework.transaction.annotation.Transactional;
6174     import org.springframework.web.bind.annotation.ModelAttribute;
6175     import org.springframework.web.bind.annotation.RequestMapping;
6176     import org.springframework.web.bind.annotation.RequestMethod;
6177     import org.springframework.web.servlet.ModelAndView;
6178
6179     import com.example.biz.dao.UserRepository;
6180     import com.example.biz.vo.User;
6181
6182     @Controller
6183     public class HelloController {
6184
6185         @Autowired
6186         UserRepository repository;
6187
6188         @RequestMapping(value = "/", method = RequestMethod.GET)
6189         public ModelAndView index(@ModelAttribute("formModel") User mydata, ModelAndView
        mav) {
6190             mav.setViewName("index");
6191             mav.addObject("msg", "this is sample content.");
6192             Iterable<User> list = repository.findAll();
6193             mav.addObject("datalist", list);
6194             return mav;
6195         }
6196
6197         @RequestMapping(value = "/", method = RequestMethod.POST)
6198         @Transactional(readonly = false)
6199         public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
        mav) {
6200             repository.saveAndFlush(mydata);
6201             return new ModelAndView("redirect:/");
6202         }
6203     }
6204
6205
6206 9. @ModelAttribute와 data 저장
6207 1)@ModelAttribute
6208     -이것은 entity class의 instance를 자동으로 적용할 때 사용
6209     -인수에는 instance 이름을 지정한다.
6210     -이것은 전송 form에서 th:object로 지정하는 값이 된다.
6211     -전송된 form의 값이 자동으로 User instance로 저장된다.
6212     -따라서 이 annotation을 이용하면 이렇게 쉽게 전송한 data를 저장할 수 있다.
6213
6214 2)saveAndFlush() method
6215     -HomeController.java의 아래 code를 보자.
```



```

6216     @RequestMapping(value = "/", method = RequestMethod.POST)
6217     @Transactional(readOnly=false)
6218     public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
        mav) {
6219         repository.saveAndFlush(mydata);
6220         return new ModelAndView("redirect:/");
6221     }
6222

```

3) 미리 설정한 entity는 JpaRepository의 saveAndFlush라는 method를 통해 entity를 영구화한다.

4) Database를 사용하고 있다면 Database에 그대로 저장된다

6225
6226

6227 10. @Transactional과 transaction

6228 1) 바로 위의 code에서 @Transactional(readOnly=false)가 있다.

6229 2) 이 annotation은 transaction을 위한 것이다.

6230 3) 이 annotation때문에 method내에서 실행되는 database 처리가 일괄적으로 실행되게 된다.

6231 4) data 변경 처리는 도중에 외부 접속에 의해 data 구조나 내용이 바뀌면 data 일관성에 문제가 발생하게 된다.

6232 5) 이런 문제를 방지하기 위해 transaction이 사용되는 것이다

6233 6) code를 보면 readOnly=false라고 설정하고 있다.

6234 7) 이 readOnly는 문자 그대로 '읽기 전용(변경 불가)'임을 의미한다.

6235 8) readOnly=false라고 설정하면 변경을 허가하는 transaction이다.

6236
6237

6238 11. Data 초기화 처리

6239 1) 저장한 data는 application을 종료하고 다시 실행하면 지워진다.

6240 2) HSQLDB는 기본적으로 memory내에 data를 cache하고 있으므로 종료와 함께 지워지는 것이다.

6241 3) controller에 data를 작성하는 초기화 처리를 넣기로 한다.

6242 4) HelloController.java code 추가

6243
6244

```

6244     @PostConstruct
6245     public void init(){
6246         User user1 = new User();
6247         user1.setUsername("한지민");
6248         user1.setAge(24);
6249         user1.setEmail("javaexpert@nate.com");
6250         user1.setMemo("Hello, Spring JPA");
6251         repository.saveAndFlush(user1);
6252
6253         User user2 = new User();
6254         user2.setUsername("조용필");
6255         user2.setAge(66);
6256         user2.setEmail("aaa@aaa.com");
6257         user2.setMemo("Good Morning!");
6258         repository.saveAndFlush(user2);
6259
6260         User user3 = new User();
6261         user3.setUsername("이미자");
6262         user3.setAge(70);
6263         user3.setEmail("bbb@bbb.com");
6264         user3.setMemo("Spring Boot is very good.");
6265         repository.saveAndFlush(user3);
6266     }
6267

```

6268
6269

5) @PostConstruct는 생성자를 통해 instance가 생성된 후에 호출되는 method임을 나타낸다.

6) Controller는 처음에 한 번만 instance를 만들고 이후에는 해당 instance를 유지한다.

6270 7) 따라서 여기에 test용 data 작성 처리를 해두면 application 실행시에 반드시 한 번 실행되어, data가 준비되는 것이다.

6271

6272

6273 12. User Find 및 Update 처리하기

6274 1)src/main/resources/templates > right-click > New > Other > Web > HTML File > Next

6275 2)File name : edit.html > Finish

6276

6277 <!DOCTYPE html>

6278 <html xmlns:th="http://www.thymeleaf.org">

6279 <head>

6280 <meta charset="UTF-8">

6281 <title>edit page</title>

6282 <style>

6283 h1 {

6284 font-size: 18pt;

6285 font-weight: bold;

6286 color: gray;

6287 }

6288

6289 body {

6290 font-size: 13pt;

6291 color: gray;

6292 margin: 5px 25px;

6293 }

6294

6295 tr {

6296 margin: 5px;

6297 }

6298

6299 th {

6300 padding: 5px;

6301 color: white;

6302 background: darkgray;

6303 }

6304

6305 td {

6306 padding: 5px;

6307 color: black;

6308 background: #e0e0ff;

6309 }

6310 </style>

6311 </head>

6312 <body>

6313 <h1 th:text="\${title}">Edit page</h1>

6314 <table>

6315 <form method="post" action="/edit" th:object="\${formModel}">

6316 <input type="hidden" name="id" th:value="*{id}" />

6317 <tr>

6318 <td><label for="username">이름</label></td>

6319 <td><input type="text" name="username" th:value="*{username}" /></td>

6320 </tr>

6321 <tr>

6322 <td><label for="age">연령</label></td>

6323 <td><input type="text" name="age" th:value="*{age}" /></td>

6324 </tr>

6325 <tr>

6326 <td><label for="email">메일</label></td>

6327 <td><input type="text" name="email" th:value="*{email}" /></td>

6328 </tr>

6329 <tr>

```

6330         <td><label for="memo">메모</label></td>
6331         <td><textarea name="memo" th:text="*{memo}" cols="20"
        rows="5"></textarea></td>
6332     </tr>
6333     <tr>
6334         <td></td>
6335         <td><input type="submit" /></td>
6336     </tr>
6337 </form>
6338 </table>
6339 </body>
6340 </html>

```

3) UserRepository code 추가

```

6341
6342 package com.example.biz.dao;
6343
6344 import java.util.Optional;
6345
6346 import org.springframework.data.jpa.repository.JpaRepository;
6347 import org.springframework.stereotype.Repository;
6348
6349 import com.example.biz.vo.User;
6350
6351 @Repository("repository")
6352 public interface UserRepository extends JpaRepository<User, Long> {
6353     public Optional<User> findById(Long id);
6354 }
6355
6356
6357

```

4) RequestHandler 작성하기

-HelloController.java code 추가

```

6358
6359 @RequestMapping(value = "/edit/{id}", method = RequestMethod.GET)
6360 public ModelAndView edit(@ModelAttribute User user, @PathVariable int id,
6361 ModelAndView mav) {
6362     mav.setViewName("edit");
6363     mav.addObject("title", "edit mydata.");
6364     Optional<User> findUser = repository.findById((long) id);
6365     mav.addObject("formModel", findUser.get());
6366     return mav;
6367 }
6368
6369
6370 @RequestMapping(value = "/edit", method = RequestMethod.POST)
6371 @Transactional(readOnly = false)
6372 public ModelAndView update(@ModelAttribute User user, ModelAndView mav) {
6373     repository.saveAndFlush(user);
6374     return new ModelAndView("redirect:/");
6375 }
6376

```

5) 접속해서 아래와 같이 URL을 입력하면

<http://localhost:8080/edit/1>

```

6380
6381 -해당 ID의 data가 표시된다.
6382 -data를 변경하고 전송해보자.
6383 -findById는 어디서 구현되는 것일까?
6384 --repository는 method의 이름을 기준으로 entity 검색 처리를 자동 생성한다.
6385 --즉 repository에 method 선언만 작성하고, 구체적인 처리를 구현할 필요가 없다.

```

```
6386
6387
6388 13. Entity delete 구현하기
6389 1)update를 하고 select를 했으니 이번에는 delete를 해 보자.
6390 2)delete.html template를 작성한다.
6391 3)src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
6392 4)File name : delete.html > Finish
6393
6394 <!DOCTYPE html>
6395 <html xmlns:th="http://www.thymeleaf.org">
6396 <head>
6397 <meta charset="UTF-8">
6398 <title>delete page</title>
6399 <style>
6400 h1 {
6401     font-size: 18pt;
6402     font-weight: bold;
6403     color: gray;
6404 }
6405
6406 body {
6407     font-size: 13pt;
6408     color: gray;
6409     margin: 5px 25px;
6410 }
6411
6412 td {
6413     padding: 0px 20px;
6414     background: #eee;
6415 }
6416 </style>
6417 </head>
6418 <body>
6419     <h1 th:text="${title}">Delete page</h1>
6420     <table>
6421         <form method="post" action="/delete" th:object="${formModel}">
6422             <input type="hidden" name="id" th:value="*{id}" />
6423             <tr>
6424                 <td><p th:text="|이름 : *{username}|"></p></td>
6425             </tr>
6426             <tr>
6427                 <td><p th:text="|연령 : *{age}|"></p></td>
6428             </tr>
6429             <tr>
6430                 <td><p th:text="*{email}"></p></td>
6431             </tr>
6432             <tr>
6433                 <td><p th:text="*{memo}"></p></td>
6434             </tr>
6435             <tr>
6436                 <td><input type="submit" value="delete" /></td>
6437             </tr>
6438         </form>
6439     </table>
6440 </body>
6441 </html>
6442
6443 5)RequestHandler 작성
```

```
6444
6445     @RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
6446     public ModelAndView delete(@PathVariable int id, ModelAndView mav) {
6447         mav.setViewName("delete");
6448         mav.addObject("title", "delete mydata.");
6449         Optional<User> user = repository.findById((long) id);
6450         mav.addObject("formModel", user.get());
6451         return mav;
6452     }
6453
6454     @RequestMapping(value = "/delete", method = RequestMethod.POST)
6455     @Transactional(readOnly = false)
6456     public ModelAndView remove(@RequestParam long id, ModelAndView mav) {
6457         repository.deleteById(id);
6458         return new ModelAndView("redirect:/");
6459     }
6460
6461 6)접속해서 실행
6462 http://localhost:8080/delete/2라고 하면 id가 2번이 출력되고 delete button을 누르면 삭제된다.
```