

- 1 Refer to <https://gist.github.com/ihoneymon/8a905e1dd8393b6b9298>
- 2 <https://www.concretepage.com/spring-boot/spring-boot-thymeleaf-maven-example>
- 3 <https://medium.com/@trevormydata/week-5-thymeleaf-with-spring-mvc-rapid-introduction-to-the-essentials-799f1fba8c07>
- 4 쇼다 츠야노 저, 김완섭 역, '스프링부터 프로그래밍 입문', (주)도서출판 길벗, 2017:4, 서울
- 5 1. 소개
- 6 1)Spring Framework는 시간이 지나면서 하위 project가 점점 늘어나서 방대해졌기 때문에 그중에서 무엇을 사용해야 할지 알수 없다는 의견이 많았다.
- 7 2)실제로 각 project를 조합해서 사용하려면 초기에 설정할 것도 많고 조합하는 방법 자체에도 knowhow가 필요했다.
- 8 3)하지만, Spring MVC를 사용하기 위해서는 필요한 framework들과 library를 정확하게 설정하지 않으면 안된다.
- 9 4)또한, 기본적인 처리를 구축하기 위해 MVC의 각 code를 작성해야 한다.
- 10 5)즉, 실질적인 프로그래밍에 들어가기 전에 하는 작업이 매우 복잡하다.
- 11 6)그래서 좀 더 효율적으로, 최소한의 작업만으로 Spring MVC를 사용한 web application을 개발할 수 없을까 하는 고민에서 탄생한 것.
- 12 7)이때 등장한 것이 Spring Boot이다.
- 13 8)덕분에 개발자는 적은 양의 application code를 작성해서 바로 실행할 수 있다.
- 14 9)실운영 server에서 사용할 수 있는 web application을 최소한의 작업으로 개발할 수 있게 설계됐다.
- 15 10)기존의 수 많은 XML 설정 file을 이용하는 Java EE 개발 방식에서 '설정 file을 사용하지 않고 annotation을 사용' 하는 방식으로 방향을 바꾸었다.
- 16 11)또한 Java EE의 세계에서는 비슷한 처리를 하는 비슷한 code를 반복해서 작성하는 경우가 많았지만 이것을 'code를 작성하지 않고 처리를 구현'하는 방식으로 변경했다.
- 17 12)그래서 실제로 Database 관련 처리는 class만 정의하면 method의 정의 없이 구현이 가능하다.
- 18 13)결론은 Spring boot는 Spring MVC를 대체하는 것이 아니라 Spring MVC를 좀 더 편하게 사용하도록 만들어 주는 도구이다.
- 19 14)Spring Boot는 Dropwizard(<https://www.dropwizard.io>)에서 영향을 받아 개발되었다.
- 20 15)Spring Boot를 이용하는 application은 Groovy 계열과 Java 계열로 크게 나누어 생각할 수 있다.
- 21 -Groovy를 이용한 application
- 22 --본격적으로 개발에 들어가기에 앞서 prototype 등을 빠르게 개발해야 하는 경우에 도움
- 23 --단시간에 application의 틀을 만들어서 보여주고 그것을 바탕으로 본격적인 개발에 들어가는 것
- 24 -Java를 이용한 application
- 25 --Maven or Gradle이라는 build tool을 사용하는 project로 생성되며 Java를 사용해 code한다.
- 26
- 27 2. 기능
- 28 1)Create stand-alone Spring applications
- 29 2)Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- 30 3)Provide opinionated 'starter' component to simplify your build configuration
- 31 4)Automatically configure Spring whenever possible
- 32 5)Provide production-ready features such as metrics, health checks and externalized configuration
- 33 6)Absolutely no code generation and no requirement for XML configuration
- 34
- 35
- 36 3. Spring Boot 시작하기
- 37 1)Maven으로 Template Project 시작하기
- 38 2)Groovy로 application 개발하기
- 39 3)<http://start.spring.io>에서 생성하기
- 40 4)STS로 project 만들기
- 41
- 42
- 43 4. Lab
- 44 1)Template Project 생성하기
- 45 -Maven으로 project template 생성하기
- 46 \$ cd SpringHome
- 47

```

48 $ mvn -B archetype:generate -DgroupId=com.example -DartifactId=springbootdemo
49 -Dversion=1.0.0-SNAPSHOT -DarchetypeArtifactId=maven-archetype-quickstart
50 [INFO] Scanning for projects...
51 ...
52 [INFO] -----
53 [INFO] Using following parameters for creating project from Old (1.x) Archetype:
    maven-archetype-quickstart:1.0
54 [INFO] -----
55 [INFO] Parameter: basedir, Value: C:\SpringHome
56 [INFO] Parameter: package, Value: com.example
57 [INFO] Parameter: groupId, Value: com.example
58 [INFO] Parameter: artifactId, Value: springbootdemo
59 [INFO] Parameter: packageName, Value: com.example
60 [INFO] Parameter: version, Value: 1.0.0-SNAPSHOT
61 [INFO] project created from Old (1.x) Archetype in dir: C:\SpringHome\springbootdemo
62 [INFO] -----
63 [INFO] BUILD SUCCESS
64 [INFO] -----
65 [INFO] Total time: 08:37 min
66 [INFO] Finished at: 2018-06-24T22:23:10+09:00
67 [INFO] -----
68 <[0m<[0m

```

-springbootdemo folder로 이동

```

72 $ cd springbootdemo
73 |
74 |----src
75 |
76 |----main
77 |
78 |----java
79 |
80 |----com
81 |
82 |----example
83 |
84 |----App.java
85 |
86 |----test

```

2)pom.xml 설정하기

```

90 <?xml version="1.0" encoding="utf-8" ?>
91 <project xmlns="http://maven.apache.org/POM/4.0.0"
92   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
93   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
94     http://maven.apache.org/maven-v4_0_0.xsd">
95   <modelVersion>4.0.0</modelVersion>
96   <groupId>com.example</groupId>
97   <artifactId>springbootdemo</artifactId>
98   <packaging>jar</packaging>
99   <version>1.0.0-SNAPSHOT</version>

```

```
98     <name>springbootdemo</name>
99     <url>http://maven.apache.org</url>
100
101     <parent>
102         <groupId>org.springframework.boot</groupId>
103         <artifactId>spring-boot-starter-parent</artifactId>
104         <version>1.5.14.RELEASE</version>
105     </parent>
106
107     <dependencies>
108         <dependency>
109             <groupId>junit</groupId>
110             <artifactId>junit</artifactId>
111             <version>4.12</version>
112             <scope>test</scope>
113         </dependency>
114         <dependency>
115             <groupId>org.springframework.boot</groupId>
116             <artifactId>spring-boot-starter-web</artifactId>
117         </dependency>
118         <dependency>
119             <groupId>org.springframework.boot</groupId>
120             <artifactId>spring-boot-starter-test</artifactId>
121             <scope>test</scope>
122         </dependency>
123     </dependencies>
124
125     <build>
126         <plugins>
127             <plugin>
128                 <groupId>org.springframework.boot</groupId>
129                 <artifactId>spring-boot-maven-plugin</artifactId>
130             </plugin>
131         </plugins>
132     </build>
133
134     <properties>
135         <java.version>1.8</java.version>
136     </properties>
137 </project>
138
139 -<parent> 설정하기
140 --Spring Boot의 설정 정보를 상속한다.
141 -- 여기서 지정한 version이 spring boot의 version이 된다.
142 --spring boot의 version을 올리려면 <version> tag 안에 있는 설정 값을 변경한다.
143
144 -spring-boot-starter-web
145 --spring boot로 web application을 만들 때 참조할 기본 library 정보를 설정한다.
146 --이렇게 쓰기만 해도 web application 제작에 필요한 spring framework 관련 library와 third-party
147 library를 이용할 수 있게 된다.
148 --version은 위 parent에서 설정한 spring-boot-starter-parent 안에 정의되어 있으므로, 여기서는 지정하지
149 않아도 된다.
```

3)사용가능한 library 확인

```

150
151 $ mvn dependency:tree
152 ...
153 [INFO] com.example:springbootdemo:jar:1.0.0-SNAPSHOT
154 [INFO] +- junit:junit:jar:3.8.1:test
155 [INFO] +- org.springframework.boot:spring-boot-starter-web:jar:1.5.14.RELEASE:compile
156 [INFO] | +- org.springframework.boot:spring-boot-starter:jar:1.5.14.RELEASE:compile
157 [INFO] | | +- org.springframework.boot:spring-boot:jar:1.5.14.RELEASE:compile
158 [INFO] | | +-
org.springframework.boot:spring-boot-autoconfigure:jar:1.5.14.RELEASE:compile
159 [INFO] | | +-
org.springframework.boot:spring-boot-starter-logging:jar:1.5.14.RELEASE:compile
160 [INFO] | | | +- ch.qos.logback:logback-classic:jar:1.1.11:compile
161 [INFO] | | | \- ch.qos.logback:logback-core:jar:1.1.11:compile
162 [INFO] | | +- org.slf4j:jcl-over-slf4j:jar:1.7.25:compile
163 [INFO] | | +- org.slf4j:jul-to-slf4j:jar:1.7.25:compile
164 [INFO] | | \- org.slf4j:log4j-over-slf4j:jar:1.7.25:compile
165 [INFO] | \- org.yaml:snakeyaml:jar:1.17:runtime
166 [INFO] +-
org.springframework.boot:spring-boot-starter-tomcat:jar:1.5.14.RELEASE:compile
167 [INFO] | +- org.apache.tomcat.embed:tomcat-embed-core:jar:8.5.31:compile
168 [INFO] | | \- org.apache.tomcat:tomcat-annotations-api:jar:8.5.31:compile
169 [INFO] | +- org.apache.tomcat.embed:tomcat-embed-el:jar:8.5.31:compile
170 [INFO] | \- org.apache.tomcat.embed:tomcat-embed-websocket:jar:8.5.31:compile
171 [INFO] +- org.hibernate:hibernate-validator:jar:5.3.6.Final:compile
172 [INFO] | +- javax.validation:validation-api:jar:1.1.0.Final:compile
173 [INFO] | +- org.jboss.logging:jboss-logging:jar:3.3.2.Final:compile
174 [INFO] | \- com.fasterxml:classmate:jar:1.3.4:compile
175 [INFO] +- com.fasterxml.jackson.core:jackson-databind:jar:2.8.11.2:compile
176 [INFO] | +- com.fasterxml.jackson.core:jackson-annotations:jar:2.8.0:compile
177 [INFO] | \- com.fasterxml.jackson.core:jackson-core:jar:2.8.11:compile
178 [INFO] +- org.springframework:spring-web:jar:4.3.18.RELEASE:compile
179 [INFO] | +- org.springframework:spring-aop:jar:4.3.18.RELEASE:compile
180 [INFO] | +- org.springframework:spring-beans:jar:4.3.18.RELEASE:compile
181 [INFO] | \- org.springframework:spring-context:jar:4.3.18.RELEASE:compile
182 [INFO] \- org.springframework:spring-webmvc:jar:4.3.18.RELEASE:compile
183 [INFO] \- org.springframework:spring-expression:jar:4.3.18.RELEASE:compile
184 [INFO] \- org.springframework.boot:spring-boot-starter-test:jar:1.5.14.RELEASE:test
185 [INFO] +- org.springframework.boot:spring-boot-test:jar:1.5.14.RELEASE:test
186 [INFO] +-
org.springframework.boot:spring-boot-test-autoconfigure:jar:1.5.14.RELEASE:test
187 [INFO] +- com.jayway.jsonpath:json-path:jar:2.2.0:test
188 [INFO] | +- net.minidev:json-smart:jar:2.2.1:test
189 [INFO] | | \- net.minidev:accessors-smart:jar:1.1:test
190 [INFO] | | \- org.ow2.asm:asm:jar:5.0.3:test
191 [INFO] | \- org.slf4j:slf4j-api:jar:1.7.25:compile
192 [INFO] +- org.assertj:assertj-core:jar:2.6.0:test
193 [INFO] +- org.mockito:mockito-core:jar:1.10.19:test
194 [INFO] | \- org.objenesis:objenesis:jar:2.1:test
195 [INFO] +- org.hamcrest:hamcrest-core:jar:1.3:test
196 [INFO] +- org.hamcrest:hamcrest-library:jar:1.3:test
197 [INFO] +- org.skyscreamer:jsonassert:jar:1.4.0:test
198 [INFO] | \- com.vaadin.external.google:android-json:jar:0.0.20131108.vaadin1:test
199 [INFO] +- org.springframework:spring-core:jar:4.3.18.RELEASE:compile

```

```

200 [INFO] \- org.springframework:spring-test:jar:4.3.18.RELEASE:test
201 [INFO] -----
202 [INFO] BUILD SUCCESS
203 [INFO] -----
204 [INFO] Total time: 10:09 min
205 [INFO] Finished at: 2018-06-24T23:07:52+09:00
206 [INFO] -----

```

208 4)Hello World!를 출력하는 Web application 작성하기

209 -src/main/java/com/example/App.java

```

210
211 package com.example;

```

```

212
213 import org.springframework.boot.SpringApplication;
214 import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
215 import org.springframework.web.bind.annotation.RequestMapping;
216 import org.springframework.web.bind.annotation.RestController;

```

```

217
218 @RestController
219 @EnableAutoConfiguration
220 public class App {

```

```

221
222     @RequestMapping("/")
223     String home(){
224         return "Hello, World!";
225     }

```

```

226
227     public static void main( String[] args ){
228         SpringApplication.run(App.class, args);
229     }
230 }

```

232 [-@RestController](#)

233 --이 annotation을 붙이면 web appication에서 request를 받아들이는 controller class임을 나타낸다.

235 [-@EnableAutoConfiguration](#)

236 --이 annotation은 매우 중요하다.

237 --이 annotation을 붙이면 다양한 설정이 자동으로 수행되고 기존의 spring application에 필요했던 설정 file들이 필요없게 된다.

239 [-@RequestMapping\("/"\)](#)

240 --이 annotation이 붙으면 이 method가 HTTP 요청을 받아들이는 method임을 나타낸다.

242 -return "Hello World!";

243 --HTTP 응답을 반환한다.

244 [-@RestController](#) annotation이 붙은 class에 속한 method에서 문자열을 반환하면 해당 문자열이 그대로 HTTP 응답이 되어 출력된다.

246 -SpringApplication.run(App.clas, args);

247 --spring boot applicaton을 실행하는 데 필요한 처리를 main() 안에서 작성한다.

248 [-@EnableAutoConfiguration](#) annotation이 붙은 class를 SpringApplication.run()의 첫번째 인자로 지정한다.

250 5)Web Application 실행하기

```

251 $ mvn spring-boot:run
252 ...
253
254
255 . _ _ _ _ _
256 /\ / _ _ _ _ _ ( ) _ _ _ _ _ \ \ \ \
257 ( ( ) \ _ _ | ' _ | ' _ | ' V _ ' | \ \ \ \
258 \ V _ _ ) | _ | | | | | | ( _ | ) ) ) )
259 ' | _ _ | . _ | | | _ | | \ _ , | / / / /
260 =====|_|=====|_____/=/_/_/_/_/
261 :: Spring Boot ::      (v1.5.14.RELEASE)
262 ...
263 ...
264 2018-06-24 23:19:10.688 INFO 31008 --- [           main] com.example.App
    : Started App in 2.191 seconds (JVM running for 70.248)

```

265
266
267 -출력된 log 내용을 보면 8080 port로 tomcat이 시작된다는 것을 알 수 있다.
268 -SpringApplication.run() method에서 내장 server를 시작했기 때문이다.
269 -<http://localhost:8080/>로 접속해보자.
270 -Web browser에 'Hello, World!'가 출력된다.
271 -application을 끝내려면 Ctrl + C를 누르고, '[일괄 작업을 끝내시겠습니까 (Y/N)]?'라는 질문에 'y'를 입력하고 enter key를 누르면 된다.

272 -여기서 알게 된 사실
273 --설정할 의존 관계의 갯수가 적다.
274 --Java Class 하나만 작성하면 된다.
275 --명령 prompt에서 application을 실행한다.

276 6)실행 가능한 jar file 작성하기

```

277 $ mvn package
278
279
280
281 -target folder 안에 springbootdemo-1.0.0-SNAPSHOT.jar file이 생성되어 있다.
282 -실행해 보자.
283
284     $ cd target
285     $ java -jar springbootdemo-1.0.0-SNAPSHOT.jar
286
287 -위와 동일하게 실행된다.
288 -이 jar file만 있으면 application 배포가 가능하다.
289 -HTTP Port도 변경할 수 있다.
290     --실행할 때 아래의 option을 붙이면 된다.
291
292     --server.port=PORT 번호
293

```

294 7)STS로 spring boot application 개발하기

- File > Import > Maven > Existing Maven Projects > Next
- Click [Browse...] > Select [springbootdemo] > Finish
- src/main/java/com.example.App.java > right-click > Run As > Spring Boot App

298 8)STS로 spring boot project 생성하기

- Package Explorer > right-click > New > Spring Starter Project
- Service URL : <http://start.spring.io>
- Name : demo

```
303 -Type : Maven
304 -Packaging : jar
305 -Java Version : 8
306 -Language : Java
307 -Group : com.example
308 -Artifact : demo
309 -Version : 0.0.1-SNAPSHOT
310 -Description : Demo project for Spring Boot
311 -Package : com.example.demo
312
313 -Next >
314 -Spring Boot Version : 1.5.15 (SNAPSHOT)
315 -Select Web > check Web > Finish
316
317 -src/main/java/com.example.demo.DemoApplication.java 수정하기
318
319 package com.example.demo;
320
321 import org.springframework.boot.SpringApplication;
322 import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
323 import org.springframework.web.bind.annotation.RequestMapping;
324 import org.springframework.web.bind.annotation.RestController;
325
326 @RestController
327 @EnableAutoConfiguration
328 public class DemoApplication {
329
330     @RequestMapping("/")
331     String home() {
332         return "Hello, World!";
333     }
334
335     public static void main(String[] args) {
336         SpringApplication.run(DemoApplication.class, args);
337     }
338 }
339
340 -DemoApplication.java > right-click > Run As > Spring Boot App
341 -http://localhost:8080/
342
343
344 9)Spring loaded로 간편하게 개발하기
345 -spring loaded를 사용하면 Java code를 수정했을 때 application을 실행한 상태에서 수정한 code의 내용을
    application에 반영할 수 있다.
346 -이러한 기능을 'Hot Reloading'이라고 부른다.
347 -spring loaded를 사용하기 위해 pom.xml에 code를 추가한다.
348
349 <build>
350 <plugins>
351 <plugin>
352 <groupId>org.springframework.boot</groupId>
353 <artifactId>spring-boot-maven-plugin</artifactId>
354 <!-- 여기부터 시작 -->
355 <dependencies>
```

```

356         <dependency>
357             <groupId>org.springframework</groupId>
358             <artifactId>springloaded</artifactId>
359             <version>1.2.3.RELEASE</version>
360         </dependency>
361     </dependencies>
362     <!-- 여기까지 -->
363 </plugin>
364 </plugins>
365 </build>
366
367 -command prompt에서
368   $ cd demo
369   $ mvn spring-boot:run
370
371 -http://localhost:8080/
372   Hello, World!
373
374 -DemoApplication.java 수정
375
376   String home() {
377       return "Hello Spring Boot!";
378   }
379
380 -Save 하고 다시 http://localhost:8080으로 가면 자동으로 변경되어
381   Hello Spring Boot!";
382
383
384 5. Groovy로 application 개발하기
385 1)준비
386   -Visit
387     https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html
388   -10.2.1 Manual Installation에서 spring-boot-cli-2.0.3.RELEASE-bin.zip link를 click한다.
389   -하지만 여기서는 1.5.3을 download하기 위해
390     https://repo.spring.io/release/org/springframework/boot/spring-boot-cli/1.5.3.RELEASE/를 방문한다.
391   -Download spring-boot-cli-1.5.3.RELEASE-bin.zip
392   -Unzip > Move to C:\Program Files\spring-1.5.3.RELEASE
393   -path 설정
394     --%PATH%;C:\Program Files\spring-1.5.3.RELEASE\bin
395
396 2)Groovy Script 작성하기
397   -Editor를 열어서 아래의 code를 적당한 위치(즉 C:\temp)에 file 이름은 app.groovy라고 저장한다.
398
399   --app.groovy
400
401   @RestController
402   class App {
403
404       @RequestMapping("/")
405       def home() {
406           "Hello!!!"
407       }
408   }

```


Page 9 of 53

```

437 2018-06-25 23:36:34.145 INFO 27432 --- [runner-0]
s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice:
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationCo
438 2018-06-25 23:36:34.213 INFO 27432 --- [runner-0]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/]}" onto public
java.lang.Object App.home()
439 2018-06-25 23:36:34.217 INFO 27432 --- [runner-0]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public
org.springframework.http.ResponseEntity<java.util.Map<java.lang.String,
java.lang.Object>>
org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.
HttpServletRequest)
440 2018-06-25 23:36:34.218 INFO 27432 --- [runner-0]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}"
onto public org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.
http.HttpServletRequest,javax.servlet.http.HttpServletResponse)
441 2018-06-25 23:36:34.256 INFO 27432 --- [runner-0]
o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler
of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
442 2018-06-25 23:36:34.256 INFO 27432 --- [runner-0]
o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
443 2018-06-25 23:36:34.305 INFO 27432 --- [runner-0]
o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto
handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
444 2018-06-25 23:36:34.683 INFO 27432 --- [runner-0]
o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
445 2018-06-25 23:36:34.745 INFO 27432 --- [runner-0]
s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
446 2018-06-25 23:36:34.750 INFO 27432 --- [runner-0] o.s.boot.SpringApplication
: Started application in 3.006 seconds (JVM running for 37.809)

```

4) Web browser로 <http://localhost:8080>에 접속한다.

Hello!!!

5) app.groovy code 수정하기

-Command Prompt 에서 Ctrl + C를 눌러서 종료시킨다.

-일괄 작업을 끝내시겠습니까 (Y/N)? Y

-아래와 같이 code를 수정한다.

```
@RestController
```

```
class App {
```

```
    @RequestMapping("/")
```

```
    def home() {
```

```
        def header = "<html><body>"
```

```
        def footer = "</body></html>"
```

```
        def content = "<h1>Hello! Spring Boot with Groovy</h1><p>This is html
content.</p>"
```

```
466         header + content + footer
467     }
468 }
469
470 -다시 script를 실행한다.
471
472     $ spring run app.groovy
473
474 -Browser를 refresh 한다.
475
476     Hello! Spring Boot with Groovy
477     This is html content.
478
479 6)Template 사용하기
480 -template은 HTML을 기반으로 작성된 code를 읽어 rendering해서 web page에 출력하는 기능이다.
481 -이런 기능의 template이 몇 가지 종류가 있지만, spring boot에서는 thymeleaf(타임리프)라고 하는 library를
    자주 사용한다.
482 -http://www.thymeleaf.org
483 -template file 작성
484
485     --C:\temp\templates\home.html
486
487     <!doctype html>
488     <html lang="en">
489         <head>
490             <meta charset="UTF-8" /> <!--반드시 종결 tag 필요 -->
491             <title>Index Page</title>
492             <style type="text/css">
493                 h1 { font-size:18pt; font-weight:bold; color:gray; }
494                 body { font-size:13pt; color:gray; margin:5px 25px; }
495             </style>
496         </head>
497         <body>
498             <h1>Hello! Spring Boot with Thymeleaf</h1>
499             <p>This is sample web page.</p>
500         </body>
501     </html>
502
503 -template file은 controller가 있는 곳의 templates folder 안에 두어야 한다.
504 -controller 수정하기
505
506     --app.groovy
507
508     @Grab("thymeleaf-spring4")
509
510     @Controller
511     class App {
512
513         @RequestMapping("/")
514         @ResponseBody
515         def home(ModelAndView mav) {
516             mav.setViewName("home")
517             mav
518         }
519     }
```

```
519     }
520
521     -다시 script 실행
522
523     $ spring run app.groovy
524
525 7)form 전송하기
526
527     -home.html
528
529     <!doctype html>
530     <html lang="en">
531         <head>
532             <meta charset="UTF-8" />
533             <title>Index Page</title>
534             <style type="text/css">
535                 h1 { font-size:18pt; font-weight:bold; color:gray; }
536                 body { font-size:13pt; color:gray; margin:5px 25px; }
537             </style>
538         </head>
539         <body>
540             <h1>Hello!</h1>
541             <p th:text="${msg}">${msg}</p>
542             <form method="post" action="/send">
543                 <input type="text" name="text1" th:value="${value}" />
544                 <input type="submit" value="Send" />
545             </form>
546         </body>
547     </html>
548
549     -app.groovy
550
551     @Grab("thymeleaf-spring4")
552
553     @Controller
554     class App {
555
556         @RequestMapping(value = "/", method=RequestMethod.GET)
557         @ResponseBody
558         def home(ModelAndView mav) {
559             mav.setViewName("home")
560             mav.addObject("msg", "Please write your name...")
561             mav
562         }
563
564         @RequestMapping(value = "/send", method=RequestMethod.POST)
565         @ResponseBody
566         def send(@RequestParam("text1") String str, ModelAndView mav){
567             mav.setViewName("home")
568             mav.addObject("msg", "Hello, " + str + "!!!")
569             mav.addObject("value", str)
570             mav
571         }
572     }
```

```
573
574 -script 실행
575
576
577 6. SPRING INITIALIZR : http://start.spring.io에서 생성하기, Maven
578 1)Visit http://start.spring.io/
579 2)설정
580 -Maven Project
581 -Java
582 -1.5.15 (SNAPSHOT)
583 -Click [Switch to the full version]
584 -Group : com.example
585 -Artifact : demo
586 -Name : demo
587 -Description : Demo project for Spring Boot
588 -Package Name : com.example.demo
589 -Packaging : Jar
590 -Java Version : 8
591 -dependencies : Web
592
593 3)Click [Generate Project]
594 4)Downloads [demo.zip] : 48.7KB
595 5)Unpack to Spring workspace.
596 6)In STS, Package Explorer > right-click > Import > Maven > Existing Maven Projects > Next
597 7)Click [Browse...] > demo folder > Finish
598 8)pom.xml
599
600 <?xml version="1.0" encoding="UTF-8"?>
601 <project xmlns="http://maven.apache.org/POM/4.0.0"
602 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
603 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
604 <modelVersion>4.0.0</modelVersion>
605
606 <groupId>com.example</groupId>
607 <artifactId>demo</artifactId>
608 <version>0.0.1-SNAPSHOT</version>
609 <packaging>jar</packaging>
610
611 <name>demo</name>
612 <description>Demo project for Spring Boot</description>
613
614 <parent>
615 <groupId>org.springframework.boot</groupId>
616 <artifactId>spring-boot-starter-parent</artifactId>
617 <version>2.0.3.RELEASE</version>
618 <relativePath/> <!-- lookup parent from repository -->
619 </parent>
620
621 <properties>
622 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
623 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
624 <java.version>1.8</java.version>
625 </properties>
```

```

625
626     <dependencies>
627         <dependency>
628             <groupId>org.springframework.boot</groupId>
629             <artifactId>spring-boot-starter-web</artifactId>
630         </dependency>
631
632         <dependency>
633             <groupId>org.springframework.boot</groupId>
634             <artifactId>spring-boot-starter-test</artifactId>
635             <scope>test</scope>
636         </dependency>
637     </dependencies>
638
639     <build>
640         <plugins>
641             <plugin>
642                 <groupId>org.springframework.boot</groupId>
643                 <artifactId>spring-boot-maven-plugin</artifactId>
644             </plugin>
645         </plugins>
646     </build>
647
648 </project>
649
650 9)src/test/java/com.example.demo.DemoApplicationTests.java > right-click > Run As > JUnit
Test > Green bar
651 10)demp project > right-click > Run As > Spring Boot App
652 11)http://localhost:8080/
653
654 Whitelabel Error Page
655 This application has no explicit mapping for /error, so you are seeing this as a fallback.
656
657 Tue Jun 26 23:14:28 KST 2018
658 There was an unexpected error (type=Not Found, status=404).
659 No message available
660
661 12)Controller 생성
662 -src/main/java/com.example.demo > right-click > New > Class
663 -Name : HelloController
664
665     package com.example.demo;
666
667     import org.springframework.web.bind.annotation.GetMapping;
668     import org.springframework.web.bind.annotation.RestController;
669
670     @RestController
671     public class HelloController {
672
673         @GetMapping("/")
674         public String hello() {
675             return "Hello, Spring Boot World";
676         }
677     }

```

```
678
679 13)Relaunch demo
680 14)http://localhost:8080/
681
682
683 7. SPRING INITIALIZR : http://start.spring.io에서 생성하기, Gradle
684 1)Visit http://start.spring.io/
685 2)설정
686 -Gradle Project
687 -Java
688 -1.5.15 (SNAPSHOT)
689 -Click [Switch to the full version]
690 -Group : com.example
691 -Artifact : demoweb
692 -Name : demoweb
693 -Description : Demo project for Spring Boot
694 -Package Name : com.example.demoweb
695 -Packaging : Jar
696 -Java Version : 8
697 -dependencies : Core > DevTools, Web > Web
698
699 3)Click [Generate Project]
700 4)Downloads [demoweb.zip] : 55.6KB
701 5)Unpack to Spring workspace.
702 6)In STS, Package Explorer > right-click > Import > Gradle > Existing Gradle Project > Next >
703 Next
704 7)Click [Browse...] > demoweb folder > Next > Finish
705 8)build.gradle
706
707     buildscript {
708         ext {
709             springBootVersion = '1.5.15.BUILD-SNAPSHOT'
710         }
711         repositories {
712             mavenCentral()
713             maven { url "https://repo.spring.io/snapshot" }
714             maven { url "https://repo.spring.io/milestone" }
715         }
716         dependencies {
717             classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}")
718         }
719     }
720
721     apply plugin: 'java'
722     apply plugin: 'eclipse'
723     apply plugin: 'org.springframework.boot'
724
725     group = 'com.example'
726     version = '0.0.1-SNAPSHOT'
727     sourceCompatibility = 1.8
728
729     repositories {
730         mavenCentral()
731         maven { url "https://repo.spring.io/snapshot" }
```

```
731     maven { url "https://repo.spring.io/milestone" }
732 }
733
734
735     dependencies {
736         compile('org.springframework.boot:spring-boot-starter-web')
737         runtime('org.springframework.boot:spring-boot-devtools')
738         testCompile('org.springframework.boot:spring-boot-starter-test')
739     }
740
741 9)src/test/java/com.example.demoweb.DemowebApplicationTests.java > right-click > Run As
742 > JUnit Test > Green bar
743 9)demp project > right-click > Run As > Spring Boot App
744 10)http://localhost:8080/
745
746 Whitelabel Error Page
747 This application has no explicit mapping for /error, so you are seeing this as a fallback.
748
749 Tue Jun 26 23:14:28 KST 2018
750 There was an unexpected error (type=Not Found, status=404).
751 No message available
752
753 11)Controller 생성
754 -src/main/java/com.example.demoweb > right-click > New > Class
755 -Name : HomeController
756
757     package com.example.demo;
758
759     import org.springframework.web.bind.annotation.GetMapping;
760     import org.springframework.web.bind.annotation.RestController;
761
762     @RestController
763     public class HelloController {
764
765         @GetMapping("/")
766         public String home() {
767             return "Hello, Spring Boot World";
768         }
769     }
770
771 12)Relaunch demo
772 13)http://localhost:8080/
773
774 8. STS로 Spring Boot project : 정적 page
775 1)Spring Boot project 생성
776 -Package Explorer > right-click > New > Spring Starter Project
777 -Service URL : http://start.spring.io
778 -Name : springweb
779 -Type : Maven
780 -Packaging : jar
781 -Java Version : 8
782 -Language : Java
783 -Group : com.example
```



```
784 -Artifact : springweb
785 -Version : 0.0.1-SNAPSHOT
786 -Description : Demo project for Spring Boot
787 -Package : com.example.biz
788
789 -Next >
790 -Spring Boot Version : 1.5.15 (SNAPSHOT)
791 -Select Web > check Core > DevTools, Web > Web > Next > Finish
792
793 2)Controller 생성
794 -src/main/java/com.example.biz > right-click > New > Class
795 -Name : HomeController
796
797     package com.example.biz;
798
799     import org.springframework.stereotype.Controller;
800     import org.springframework.web.bind.annotation.GetMapping;
801     import org.springframework.web.servlet.ModelAndView;
802
803     @Controller
804     public class HomeController {
805
806         @GetMapping("/")
807         public ModelAndView home(ModelAndView mav) {
808             mav.setViewName("index.html");
809             return mav;
810         }
811     }
812
813 3)static file 생성
814 -src/main/resources/static/images folder 생성
815     --spring-boot.png 추가할 것
816 -src/main/resources/static/js folder 생성
817     --jquery-3.3.1.slim.min.js 추가할 것
818 -src/main/resources/static/css folder 생성
819     --bootstrap.min.css 추가할 것
820 -src/main/resources/static/index.html
821
822     <!DOCTYPE html>
823     <html>
824     <head>
825     <meta charset="UTF-8">
826     <title>Home page</title>
827     <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css" />
828     <script src="js/jquery-3.3.1.slim.min.js"></script>
829     <script>
830         $(document).ready(function(){
831             alert("Hello, Spring Boot World!!!");
832         });
833     </script>
834     </head>
835     <body>
836     <div>
837         
```

```
838     </div>
839     <div class="jumbotron">
840         <h1>Hello, Spring Boot World</h1>
841         <p>...</p>
842         <p><a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a></p>
843     </div>
844 </body>
845 </html>
846
847 4)Spring Boot에서는 template을 사용하지 않을 경우 기본적으로 static file은 src/main/resources/static에서
    찾는다.
848 5)이럴 때는 반드시 file의 확장자 .html까지 넣어야 한다.
849
850
851 9. STS로 Spring Boot project : JSP
852 1)Spring Boot에서는 JSP 사용을 권장하지 않는다.
853 2)'jar' 형식으로 동작하지 않고 War file로 배포해야 하는 등의 몇 가지 제약이 있어서이기도 하지만, 가장 큰 이유는 이미
    JSP 자체가 Server 측 언어로 그 사용 빈도가 줄고 있기 때문이다.
854 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.
855 4)그래서 template을 통해 code를 분리해야 할 필요가 있는 것이다.
856 5)Spring Boot project 생성
857     -Package Explorer > right-click > New > Spring Starter Project
858     -Service URL :http://start.spring.io
859     -Name : springwebdemo
860     -Type : Maven
861     -Packaging : jar
862     -Java Version : 8
863     -Language : Java
864     -Group : com.example
865     -Artifact : springwebdemo
866     -Version : 0.0.1-SNAPSHOT
867     -Description : Demo project for Spring Boot
868     -Package : com.example.biz
869
870     -Next >
871     -Spring Boot Version : 1.5.15 (SNAPSHOT)
872     -Select Web > check Web > Finish
873
874 6)pom.xml
875     -jstl을 위해
876         <dependency>
877             <groupId>javax.servlet</groupId>
878             <artifactId>jstl</artifactId>
879             <version>1.2</version>
880         </dependency>
881
882     -JSP를 위해
883         <dependency>
884             <groupId>org.apache.tomcat</groupId>
885             <artifactId>tomcat-jasper</artifactId>
886             <version>8.5.32</version>
887         </dependency>
888
```

```
889 7)folder 준비
890 -src/main folder 안에 webapp folder 생성
891 -webapp folder 안에 WEB-INF folder 생성
892 -WEB-INF folder 안에 jsp folder 생성
893
894 8)만일 JSP를 template으로 사용하는 경우에는 이 WEB-INF folder 안에 template file을 준비할 필요가 있다.
895 9)src/main/resources/application.properties code 추가
896
897 spring.mvc.view.prefix : /WEB-INF/jsp/
898 spring.mvc.view.suffix : .jsp
899
900 10)jsp folder 안에 jsp file 생성
901 -index.jsp
902
903 <%@ page language="java" contentType="text/html; charset=UTF-8"
904     pageEncoding="UTF-8"%>
905 <%@ page import="java.util.Date, java.text.SimpleDateFormat" %>
906 <!DOCTYPE html>
907 <html>
908 <head>
909 <meta charset="UTF-8">
910 <title>Insert title here</title>
911 </head>
912 <body>
913 <h1>Index page</h1>
914 <%=new SimpleDateFormat("yyyy년 MM월 dd일").format(new Date()) %>
915 </body>
916 </html>
917
918 11)com.example.biz.HomeController.java
919
920 package com.example.biz;
921
922 import org.springframework.stereotype.Controller;
923 import org.springframework.web.bind.annotation.GetMapping;
924
925 @Controller
926 public class HomeController {
927
928     @GetMapping("/")
929     public String index() {
930         return "index";
931     }
932 }
933
934 12)<a href="http://localhost:8080/">http://localhost:8080/
935
936 Index page
937 2018년 06월 27일
938
939
940 10. Lab : thymeleaf template 사용하기
941 1)Spring Boot project 생성
942 -Package Explorer > right-click > New > Spring Starter Project
```

```
943 -Service URL :http://start.spring.io
944 -Name : MyBootApplication
945 -Type : Maven
946 -Packaging : jar
947 -Java Version : 8
948 -Language : Java
949 -Group : com.example
950 -Artifact : MyBootApplication
951 -Version : 0.0.1-SNAPSHOT
952 -Description : Demo project for Spring Boot
953 -Package : com.example.biz
954
955 -Next >
956 -Spring Boot Version : 1.5.15 (SNAPSHOT)
957 -Select Web > check Web > Finish
958
```

959 2)src/main/java/com.example.biz.MyBootApplicationApplication.java 수정하기

```
960
961     package com.example.biz;
962
963     import org.springframework.boot.SpringApplication;
964     import org.springframework.boot.autoconfigure.SpringBootApplication;
965
966     @SpringBootApplication
967     public class MyBootApplicationApplication {
968
969         public static void main(String[] args) {
970             SpringApplication.run(MyBootApplicationApplication.class, args);
971         }
972     }
973
```

974 3)실행

```
975 -MyBootApplicationApplication.java > right-click > Run As > Spring Boot App
976 -http://localhost:8080/
977
978 Whitelabel Error Page
979 This application has no explicit mapping for /error, so you are seeing this as a fallback.
980
981 Tue Jun 26 09:42:08 KST 2018
982 There was an unexpected error (type=Not Found, status=404).
983 No message available
984
```

985 4)pom.xml

```
986 -parent
987     --상속에 관한 설정 정보이다.
988
989     <parent>
990         <groupId>org.springframework.boot</groupId>
991         <artifactId>spring-boot-starter-parent</artifactId>
992         <version>1.5.15.BUILD-SNAPSHOT</version>
993         <relativePath/> <!-- lookup parent from repository -->
994     </parent>
995
996 -properties
```

```
997
998     <properties>
999         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
1000         <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
1001         <java.version>1.8</java.version>
1002     </properties>
1003
1004
1005     -dependencies
1006
1007     <dependencies>
1008         <dependency>
1009             <groupId>org.springframework.boot</groupId>
1010             <artifactId>spring-boot-starter-web</artifactId>
1011         </dependency>
1012
1013         <dependency>
1014             <groupId>org.springframework.boot</groupId>
1015             <artifactId>spring-boot-starter-test</artifactId>
1016             <scope>test</scope>
1017         </dependency>
1018     </dependencies>
1019
1020     5)Controller 작성하기
1021     -com.example.biz > right-click > New > Class
1022     -Name : HelloController > Finish
1023     -HelloController.java
1024
1025     package com.example.biz;
1026
1027     import org.springframework.web.bind.annotation.RequestMapping;
1028     import org.springframework.web.bind.annotation.RestController;
1029
1030     @RestController
1031     public class HelloController {
1032
1033         @RequestMapping("/")
1034         public String index() {
1035             return "Hello Spring Boot World!";
1036         }
1037     }
1038
1039     6)project > right-click > Run As > Spring Boot App
1040     7)http://localhost:8080/
1041
1042     Hello Spring Boot World!
1043
1044     8)매개변수 전달
1045     -HelloController.java 수정
1046
1047     @RequestMapping("/{num}")
1048     public String index(@PathVariable int num) {
1049         int result = 0;
1050         for(int i = 1 ; i <= num ; i++) result += i;
```

```
1051         return "total : " + result;
1052     }
1053
1054 9)project > right-click > Run As > Spring Boot App
1055 10)http://localhost:8080/100
1056
1057     total : 5050
1058
1059 10)객체를 JSON으로 출력하기
1060     -src/main/java/com.example.biz/Student.java 생성
1061
1062     package com.example.biz;
1063
1064     public class Student {
1065         private int userid;
1066         private String name;
1067         private int age;
1068         private String address;
1069         public Student(int userid, String name, int age, String address) {
1070             this.userid = userid;
1071             this.name = name;
1072             this.age = age;
1073             this.address = address;
1074         }
1075         public int getUserid() {
1076             return userid;
1077         }
1078         public void setUserid(int userid) {
1079             this.userid = userid;
1080         }
1081         public String getName() {
1082             return name;
1083         }
1084         public void setName(String name) {
1085             this.name = name;
1086         }
1087         public int getAge() {
1088             return age;
1089         }
1090         public void setAge(int age) {
1091             this.age = age;
1092         }
1093         public String getAddress() {
1094             return address;
1095         }
1096         public void setAddress(String address) {
1097             this.address = address;
1098         }
1099     }
1100
1101 11)HelloController.java 수정
1102
1103     @RestController
1104     public class HelloController {
```

```

1105     String [] names = {"조용필", "이미자", "설운도"};
1106     int [] ages = {56, 60, 70};
1107     String [] addresses = {"서울특별시", "부산광역시", "대전광역시"};
1108
1109     @RequestMapping("/{userid}")
1110     public Student index(@PathVariable int userid) {
1111         return new Student(userid, names[userid], ages[userid], addresses[userid]);
1112     }
1113 }

```

12) <http://localhost:8080/1>

```

{"userid":1,"name":"이미자","age":60,"address":"부산광역시"}

```

13) thymeleaf 추가하기

-pom.xml 수정하기

-Select Dependencies tab > Add

-Group Id : org.springframework.boot

-Artifact Id : spring-boot-starter-thymeleaf

-Version :

-Scope : compile

-OK

```

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>

```

14) HelloController.java 수정

```

package com.example.biz;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HelloController {

    @RequestMapping("/")
    public String index() {
        return "index";
    }
}

```

15) template file 생성

-src/main/resources/templates > right-click > New > Other... > Web > HTML File > Next

-File name : index.html

```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />  <!--반드시 종결 tag 필요 -->
    <title>Index Page</title>
    <style type="text/css">
      h1 { font-size:18pt; font-weight:bold; color:gray; }

```

```
1159         body { font-size:13pt; color:gray; margin:5px 25px; }
1160     </style>
1161 </head>
1162 <body>
1163     <h1>Hello! Spring Boot with Thymeleaf</h1>
1164     <p>This is sample web page.</p>
1165 </body>
1166 </html>
```

1167
1168 16) <http://localhost:8080/>

1169
1170 Hello! Spring Boot with Thymeleaf
1171 This is sample web page.

1172
1173 17) template에 값 표시하기
1174 -index.html code 수정

```
1175  
1176     <body>
1177         <h1>Hello! Spring Boot with Thymeleaf</h1>
1178         <p class="msg" th:text="${msg}"></p>
1179     </body>
```

1180
1181 -HelloController.java 수정

```
1182  
1183 @Controller
1184 public class HelloController {
1185  
1186     @RequestMapping("/{num}")
1187     public String index(@PathVariable int num, Model model) {
1188         int result = 0;
1189         for(int i = 1 ; i <= num ; i++) result += i;
1190         model.addAttribute("msg", "total : " + result);
1191         return "index";
1192     }
1193 }
```

1194
1195 18) <http://localhost:8080/100>

1196
1197 Hello! Spring Boot with Thymeleaf
1198 total : 5050

1200
1201 19) ModelAndView class 사용하기
1202 -HelloController.java 수정

```
1203  
1204 @Controller
1205 public class HelloController {
1206  
1207     @RequestMapping("/{num}")
1208     public ModelAndView index(@PathVariable int num, ModelAndView mav) {
1209         int result = 0;
1210         for(int i = 1 ; i <= num ; i++) result += i;
1211         mav.addObject("msg", "total : " + result);
1212         mav.setViewName("index");
```



```
1213         return mav;
1214     }
1215 }
1216
1217 20) http://localhost:8080/100
1218
1219     Hello! Spring Boot with Thymeleaf
1220     total : 5050
1221
1222 21) form 사용하기
1223     -index.html 수정
1224
1225     <!doctype html>
1226     <html lang="en">
1227         <head>
1228             <meta charset="UTF-8" />
1229             <title>Index Page</title>
1230             <style type="text/css">
1231                 h1 { font-size:18pt; font-weight:bold; color:gray; }
1232                 body { font-size:13pt; color:gray; margin:5px 25px; }
1233             </style>
1234         </head>
1235         <body>
1236             <h1>Hello!</h1>
1237             <p th:text="${msg}">${msg}</p>
1238             <form method="post" action="/send">
1239                 <input type="text" name="text1" th:value="${value}" />
1240                 <input type="submit" value="Send" />
1241             </form>
1242         </body>
1243     </html>
1244
1245     -HelloController.java 수정
1246
1247     @Controller
1248     public class HelloController {
1249
1250         @RequestMapping(value = "/", method = RequestMethod.GET)
1251         public ModelAndView index(ModelAndView mav) {
1252             mav.setViewName("index");
1253             mav.addObject("msg", "Please write your name...");
1254             return mav;
1255         }
1256
1257         @RequestMapping(value = "/send", method = RequestMethod.POST)
1258         public ModelAndView send(@RequestParam("txtName") String name,
1259                                 ModelAndView mav) {
1260             mav.addObject("msg", "안녕하세요! " + name + "님!");
1261             mav.addObject("value", name);
1262             mav.setViewName("index");
1263             return mav;
1264         }
1265     }
1266
```

```
1267 22)http://localhost:8080
1268
1269 23)기타 form controller
1270 -index.html 수정
1271
1272 <!doctype html>
1273 <html lang="en">
1274 <head>
1275 <meta charset="UTF-8" />
1276 <title>Index Page</title>
1277 <style type="text/css">
1278 h1 {
1279     font-size: 18pt;
1280     font-weight: bold;
1281     color: gray;
1282 }
1283
1284 body {
1285     font-size: 13pt;
1286     color: gray;
1287     margin: 5px 25px;
1288 }
1289 </style>
1290 </head>
1291 <body>
1292 <h1>Hello!</h1>
1293 <pre th:text="{msg}">Please wait...</pre>
1294 <form method="post" action="/">
1295 <div>
1296 <input type="checkbox" id="ckeck1" name="check1" /> <label
1297     for="ckeck1">체크</label>
1298 </div>
1299 <div>
1300 <input type="radio" id="male" name="gender" value="male" /> <label
1301     for="male">남성</label>
1302 </div>
1303 <div>
1304 <input type="radio" id="female" name="gender" value="female" /> <label
1305     for="female">여성</label>
1306 </div>
1307 <div>
1308 <select name="selOs" size="4">
1309 <option>--선택--</option>
1310 <option value="Windows">windows</option>
1311 <option value="MacOS">MacOS</option>
1312 <option value="Linux">Linux</option>
1313 </select>
1314 <select name="selEditors" size="4" multiple="multiple">
1315 <option>--선택--</option>
1316 <option value="Notepad">Notepad</option>
1317 <option value="Editplus">Editplus</option>
1318 <option value="Visual Studio Code">Visual Studio Code</option>
1319 <option value="Sublime Text">Sublime Text</option>
1320 <option value="Eclipse">Eclipse</option>
```

```
1321         </select>
1322     </div>
1323     <input type="submit" value="전송" />
1324 </form>
1325 </body>
1326 </html>
1327
1328 24)HelloController.java 수정
1329
1330 @Controller
1331 public class HelloController {
1332
1333     @RequestMapping(value = "/", method = RequestMethod.GET)
1334     public ModelAndView index(ModelAndView mav) {
1335         mav.setViewName("index");
1336         mav.addObject("msg", "값을 입력후 전송버튼을 눌러주세요.");
1337         return mav;
1338     }
1339
1340     @RequestMapping(value = "/", method = RequestMethod.POST)
1341     public ModelAndView send(@RequestParam(value="check1", required=false) boolean
check1,
1342                             @RequestParam(value="gender", required=false) String gender,
1343                             @RequestParam(value="selOs", required=false) String selOs,
1344                             @RequestParam(value="selEditors", required=false) String []
selEditors,
1345                             ModelAndView mav) {
1346
1347         String result = "";
1348         try {
1349             result = "check : " + check1 +
1350                     ", gender : " + gender +
1351                     ", OS : " + selOs +
1352                     "\nEditors : ";
1353         }catch(NullPointerException ex) {}
1354         try {
1355             result += selEditors[0];
1356             for(int i = 1 ; i < selEditors.length ; i++) result += ", " + selEditors[i];
1357         }catch(NullPointerException ex) {
1358             result += "null";
1359         }
1360         mav.addObject("msg", result);
1361         mav.setViewName("index");
1362         return mav;
1363     }
1364 }
1365
1366 25)http://localhost:8080
1367
1368 26)Redirect
1369 -index.html 수정
1370
1371 <body>
1372 <h1>Hello! index.</h1>
```

```

1373     </body>
1374
1375 -HelloController.java 수정
1376
1377     @Controller
1378     public class HelloController {
1379
1380         @RequestMapping("/")
1381         public ModelAndView index(ModelAndView mav) {
1382             mav.setViewName("index");
1383             return mav;
1384         }
1385
1386         @RequestMapping("/other")
1387         public String other() {
1388             return "redirect:/";
1389         }
1390
1391         @RequestMapping("/home")
1392         public String home() {
1393             return "forward:/";
1394         }
1395     }
1396

```

-redirect와 forward 차이점 구분하기

11. Templates

1)Thymeleaf

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

2)JSP

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

3)FreeMarker

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

4)Groovy

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

1422
1423 5)Velocity
1424 -Apache Software Foundation이 개발 중인 template engine이다.
1425 -\$나 #같은 특수한 문자를 사용해서 변수 등을 HTML code 내에 직접 작성할 수 있다.
1426 -FreeMarker와 비슷하지만, FreeMarker가 tag 형태로 구문을 작성하는 것에 비해 이것은 tag 없이 바로 작성할 수 있어서 Visual Editor 등에 영향이 적다.

1427
1428 6)Mustache
1429 -다양한 언어에 적용할 수 있는 template이다.
1430 -Java 뿐만 아니라 PHP, Ruby, Python, JavaScript 등에서 사용할 수 있어서 Java 이외의 언어를 사용하고 싶은 사람에게 좋은 대안이 될 수 있다.
1431 -{{}} 기호를 사용해서 변수 등을 HTML code 내에 기술한다.

1432
1433 7)기본은 변수식과 OGNL
1434 -Thymeleaf의 기본은 '값을 출력하는(표시하는) 것'이다.
1435 -이것은 \${OO} 형식으로 작성
1436 -이 형식을 '변수식'이라고 부른다.
1437 -변수식 안에 작성하는 것은 'OGNL(Object-Graph Navigation Language)식'이다.
1438 -OGNL은 Java의 값에 접근하기 위한 식이다.
1439 -Thymeleaf뿐만 아니라 다양한 library와 framework 등에서 사용되고 있다.

1440
1441 8)Lab
1442 -Spring Boot project 생성
1443 --Package Explorer > right-click > New > Spring Starter Project
1444 --Service URL : <http://start.spring.io>
1445 --Name : templatedemo
1446 --Type : Maven
1447 --Packaging : Jar
1448 --Java Version : 8
1449 --Language : Java
1450 --Group : com.example
1451 --Artifact : templatedemo
1452 --Version : 0.0.1-SNAPSHOT
1453 --Description : Demo project for Spring Boot
1454 --Package : com.example.biz
1455
1456 --Next >
1457 --Spring Boot Version : 1.5.15 (SNAPSHOT)
1458 --Select Web > Web, Template Engines > Thymeleaf > Finish
1459

1460 -HomeController.java 생성
1461 --com.example.biz > right-click > New > Class
1462 --Name : HomeController
1463
1464 package com.example.biz;
1465
1466 import org.springframework.stereotype.Controller;
1467 import org.springframework.web.bind.annotation.GetMapping;
1468
1469 @Controller
1470 public class HomeController {
1471
1472 @GetMapping("/")
1473 public String home() {

```

1474         return "index";
1475     }
1476 }
1477
1478 -index.html 생성
1479 --src/main/resources/templates > New > Other > Web > HTML File > Next
1480 --File name : index.html
1481
1482 <!DOCTYPE html>
1483 <html>
1484 <head>
1485 <meta charset="UTF-8" />
1486 <title>Insert title here</title>
1487 </head>
1488 <body>
1489 <h1>Hello Page</h1>
1490 <p th:text="${new java.util.Date().toString()}"></p>
1491 </body>
1492 </html>
1493
1494 -실행
1495 --http://localhost:8080
1496
1497 Hello Page
1498 Thu Jun 28 22:27:22 KST 2018
1499

```

9)Utility Object

```

1501 -변수식은 이름 그대로 변수를 기술해서 그대로 출력할 수 있다.
1502 -이 변수는 이미 예제에서 확인했듯이, controller에서 값을 준비하고 그것을 template 측에 전달해서 출력하는 것이
    기본적인 흐름이다.
1503 -하지만, Java class 중에는 이런 template에서 사용 빈도가 높은 것이 있다.
1504 -이런 class를 사용할 때도 항상 controller에서 class를 변수로 준비하고...하는 것이 매우 번거로운 일이다.
1505 -그래서 Thymeleaf에서는 자주 사용되는 class를 '#이름' 이라는 상수로 정의해서 변수식 안에 직접 작성할 수 있게
    하고 있다.
1506 -이것을 utility object라고 한다.
1507 -utility object의 종류
1508 --#strings : String class의 상수
1509 --#numbers : Number class의 상수
1510 --#booleans : Boolean class의 상수
1511 --#dates : Date class의 상수
1512 --#objects : Object class의 상수
1513 --#arrays : Array class의 상수
1514 --#lists : List class의 상수
1515 --#sets : Set class의 상수
1516 --#maps : Map class의 상수
1517 -이 객체들은 class의 상수이므로 직접 class method등을 호출해서 사용할 수 있다.
1518 -단, new #dates처럼 해서는 Date instance를 만들수 없다.
1519 -#dates.OO과 같이 class field나 method 호출 등에 사용한다.
1520
1521 -Lab
1522 --index.html 수정
1523
1524 <body>
1525 <h1>Hello Page</h1>

```

```

1526     <p th:text="${#dates.format(new java.util.Date(), 'dd/MM/yyyy HH:mm')}"></p>
1527     <p th:text="${#numbers.formatInteger(1234,7)}"></p>
1528     <p th:text="${#strings.toUpperCase('Welcome to Spring.')}"></p>
1529 </body>
1530
1531 -실행
1532
1533     Hello Page
1534     28/06/2018 22:38
1535
1536     0001234
1537
1538     WELCOME TO SPRING.
1539
1540 10)매개변수에 접근하기
1541 -만일 controller를 거치지 않고 query string 같은 매개변수를 사용하고자 할때는 param 이라는 변수를 사용한다.
1542 -이것은 변수식 안에서 직접 사용할 수 있는 변수이다.
1543 -이 변수 안에 있는 매개변수명을 지정해서 추출할 수 있다.
1544 -${param.id}라고 하면 id=javaexpert의 형태로 전송된 값을 받을 수 있다.
1545 -독특하게도 이 방식은 배열 형태로 되어 있어서 배열내에서 추출해야 한다.
1546 -<body> 부분을 아래와 같이 수정한다.
1547
1548     <body>
1549         <h1>Helo page</h1>
1550         <p th:text="'from parameter... id=' + ${param.id[0]} + ',name=' +
            param.name[0]'"></p>
1551     </body>
1552
1553 -HomeController.java를 수정한다.
1554
1555     @RequestMapping("/")
1556     public ModelAndView index(ModelAndView mav) {
1557         mav.setViewName("index");
1558         return mav;
1559     }
1560
1561     @RequestMapping("/home")
1562     public String home() {
1563         return "forward:/";
1564     }
1565
1566 -그리고 id와 name을 query string으로 지정해서 접속한다.
1567 --<a href="http://localhost:8080/home/?id=javaexpert&name=Springboot">http://localhost:8080/home/?id=javaexpert&name=Springboot
1568 --결과는 아래와 같다.
1569
1570     Helo page
1571     from parameter... id=javaexpert,name=Springboot
1572
1573 -controller를 거치지 않고 template내에서 직접 전달된 값을 사용할 수 있다.
1574 -중요한 것은 param내의 id나 name 배열에서 첫 번째 요소를 지정해서 추출한다는 것이다.
1575 -그 이유는 query string으로 값을 전송할 때 같은 이름의 값을 여러 개 전송하기 위해서다.
1576 -예를 들면, <a href="http://localhost:8080/home/?id=123&id=456&name=javaexpert&name=peter">http://localhost:8080/home/?id=123&id=456&name=javaexpert&name=peter
1577 -이렇게 하면 param에서 추출하는 id와 name이 각각 {123,456}, {javaexpert,peter}가 되는 것이기 때문이
    다.

```

```

1578
1579 -여기서 사용하고 있는 th:text의 값을 보면 큰따옴표 안에 다시 작은 따옴표를 사용해서 값을 작성하고 있다.
1580 -이것은 OGNL로 text literal을 작성할 때 사용하는 방식이다.
1581 -이렇게 하면 다수의 literal을 연결할 때 큰 따옴표안에 작은 따옴표 literal을 사용할 수 있게 된다.
1582
1583     th:text="one two three"
1584     th:text="'one' + ' two ' + 'three'"
1585
1586 11)message식
1587 -${}라는 변수식 외에도 thymeleaf에서 사용할 수 있는 message식이 있다.
1588 -이것은 project에 미리 준비해둔 properties file(설정 file)에서 값을 가져와서 표시하는 것이다.
1589 -Java에서는 properties file에 미리 text(설정값)을 저장해두고, 필요에 따라 file 내의 text값을 꺼내서 사용하는
    경우가 있다.
1590 -이것을 template에서 사용할 수 있게 한 것이 message식이다.
1591 -형식은 아래와 같다.
1592     #{값 지정}
1593 -${}가 아니라 #{}로 작성하는 것이 특징이다.
1594 -Lab
1595     --src/main/resources > right-click > New > File
1596     --File name : messages.properties > Finish
1597
1598     content.title=Message sample page.
1599     content.message=This is sample message from properties.
1600
1601     --index.html 수정
1602
1603     <body>
1604     <h1 th:text="#{content.title}">Hello page</h1>
1605     <p th:text="#{content.message}"></p>
1606     </body>
1607
1608     --실행
1609
1610     Message sample page.
1611     This is sample message from properties.
1612
1613 12)Link식과 href
1614 -web page에서는 URL을 지정하는 link도 다양한 곳에 사용되며 이 link를 지정하기 위한 전용식도 존재한다.
1615
1616     @{주소}
1617
1618 -이것은 기본적으로 URL을 지정하는 속성(<a> tag의 href 등)에서 사용한다.
1619 -예를 들어, @{/index}라고 작성하면 /index로 가는 link를 지정할 수 있다.
1620 -Lab
1621     --index.html
1622
1623     <body>
1624     <h1 th:text="#{content.title}">Helo page</h1>
1625     <p><a th:href="@{/home/{orderId}(orderId=${param.id[0]})}">link</a></p>
1626     </body>
1627
1628     --접속할 때 query string에 id를 지정한다.
1629     --예를 들어 http://localhost:8080/?id=123에 접속하면 link에는 /home/123이 설정된다.
1630

```


1631 13)선택 객체와 변수식
1632 -변수식에서 객체를 사용하려고 하면 문제가 발생한다.
1633 -물론 `${object.name}`처럼 객체의 `property`나 `method`를 지정해서 작성하면 되지만 객체안에서 다수의 값이 존재하는 경우 일일이 작성하는 것은 매우 귀찮은 일이다.
1634 -이런 경우 객체를 지정해서 해당 객체 안에 있는 값을 추출할 수 있는 전용 변수식을 사용하면 된다.
1635 `*{}`
1636
1637
1638 -이 변수식은 객체를 처리하는 변수식 내부에서 사용한다.
1639
1640 `<OO th:object="${객체}">`
1641 `<XX th:text="*{property}">`
1642 `</OO>`
1643
1644 -tag에 'th:object'라는 속성을 사용해서 객체를 설정한다.
1645 -이렇게 하면 해당 tag 내부에서 `*{}`를 이용한 변수식을 사용할 수 있게 된다.
1646 -이 * 형태의 변수식에선 객체 내의 `property` 등을 이름만으로도 지정할 수 있다.
1647
1648 -Lab
1649 `--src/main/java/com.example.biz > right-click > New > Class`
1650 `--Name : Member`
1651
1652 `package com.example.biz;`
1653
1654 `public class Member {`
1655 `private int id;`
1656 `private String username;`
1657 `private int age;`
1658
1659 `public Member(int id, String username, int age) {`
1660 `this.id = id;`
1661 `this.username = username;`
1662 `this.age = age;`
1663 `}`
1664
1665 `public int getId() {`
1666 `return id;`
1667 `}`
1668 `public void setId(int id) {`
1669 `this.id = id;`
1670 `}`
1671 `public String getUsername() {`
1672 `return username;`
1673 `}`
1674 `public void setUsername(String username) {`
1675 `this.username = username;`
1676 `}`
1677 `public int getAge() {`
1678 `return age;`
1679 `}`
1680 `public void setAge(int age) {`
1681 `this.age = age;`
1682 `}`
1683 `}`

```
1684
1685 --HomeController.java 수정
1686
1687     @RequestMapping("/")
1688     public ModelAndView index(ModelAndView mav) {
1689         mav.setViewName("index");
1690         mav.addObject("msg","Current data.");
1691         Member obj = new Member(123, "javaexpert",24);
1692         mav.addObject("object",obj);
1693         return mav;
1694     }
1695
1696 --index.html 수정
1697
1698 <!DOCTYPE HTML>
1699 <html xmlns:th="http://www.thymeleaf.org">
1700 <head>
1701     <title>top page</title>
1702     <meta charset="UTF-8" />
1703     <style>
1704         h1 { font-size:18pt; font-weight:bold; color:gray; }
1705         body { font-size:13pt; color:gray; margin:5px 25px; }
1706         tr { margin:5px; }
1707         th { padding:5px; color:white; background:darkgray; }
1708         td { padding:5px; color:black; background:#e0e0ff; }
1709     </style>
1710 </head>
1711 <body>
1712     <h1 th:text="#{content.title}">Hello page</h1>
1713     <p th:text="${msg}">message.</p>
1714     <table th:object="${object}">
1715         <tr><th>ID</th><td th:text="*{id}"></td></tr>
1716         <tr><th>NAME</th><td th:text="*{username}"></td></tr>
1717         <tr><th>AGE</th><td th:text="*{age}"></td></tr>
1718     </table>
1719 </body>
1720 </html>
1721
1722 --실행
1723 --controller에서 object를 저장해둔 Member 값이 표 형태로 출력된다.
1724
1725
1726 12. Lab : Spring Boot와 JDBC 연동하기
1727 1)Spring Boot project 생성
1728 -Package Explorer > right-click > New > Spring Starter Project
1729 -Service URL :http://start.spring.io
1730 -Name : BootJdbcDemo
1731 -Type : Maven
1732 -Packaging : Jar
1733 -Java Version : 8
1734 -Language : Java
1735 -Group : com.example
1736 -Artifact : BootJdbcDemo
1737 -Version : 0.0.1-SNAPSHOT
```

```
1738 -Description : Demo project for Spring Boot
1739 -Package : com.example.biz
1740
1741 -Next >
1742 -Spring Boot Version : 1.5.15 (SNAPSHOT)
1743 -Select SQL > check MySQL, JDBC
1744 -Select Web > check Web > Finish
1745
1746 -위에서 type을 선택시 고려사항
1747 --만일 Embedded된 tomcat으로 Stand-Alone형태로 구동시키기 위한 목적이라면 Packaging Type을 jar
    로 선택한다.
1748 --war로 선택할 경우, 기존과 같이 외부의 tomcat으로 deploy 하는 구조로 만들어진다.
1749 --물론, source의 최종배포의 형태가 server의 tomcat에 deploy해야 하는 구조라면 처음부터 war로 만들어
    서 작업해도 상관없다.
1750 --jar로 선택하고, local에서 개발 및 test를 하다가 나중에 배포할 경우 war로 변경해서 배포를 할 수도 있다.
1751
1752 2)pom.xml
1753 -jstl 사용을 위한
1754
1755     <dependency>
1756         <groupId>javax.servlet</groupId>
1757         <artifactId>jstl</artifactId>
1758         <version>1.2</version>
1759     </dependency>
1760
1761 -jsper 사용을 위한
1762
1763     <dependency>
1764         <groupId>org.apache.tomcat</groupId>
1765         <artifactId>tomcat-jasper</artifactId>
1766         <version>8.5.31</version>
1767     </dependency>
1768
1769 3)src/main/resources/application.properties
1770
1771     spring.application.name=BootJDBCDemo
1772     spring.datasource.url=jdbc:mysql://localhost:3306/world
1773     spring.datasource.driver-class-name=com.mysql.jdbc.Driver
1774     spring.datasource.username=root
1775     spring.datasource.password=javamysql
1776
1777 4)Table 생성
1778
1779     CREATE TABLE world.User
1780     (
1781         username VARCHAR(20) PRIMARY KEY,
1782         age TINYINT NOT NULL
1783     );
1784
1785 5)VO object 생성
1786 -src/main/java > right-click > New > Package
1787 -Name : com.example.vo
1788 -com.example.vo > right-click > New > Class
1789 -Name : UserVO
```

```
1790
1791     package com.example.vo;
1792
1793     public class UserVO {
1794         private String username;
1795         private int age;
1796
1797         public String getUsername() {
1798             return username;
1799         }
1800         public void setUsername(String username) {
1801             this.username = username;
1802         }
1803         public int getAge() {
1804             return age;
1805         }
1806         public void setAge(int age) {
1807             this.age = age;
1808         }
1809         @Override
1810         public String toString() {
1811             return "UserVO [username=" + username + ", age=" + age + "];"
1812         }
1813     }
```

1815 6)Dao object 생성

```
1816 -src/main/java > right-click > New > Package
1817 -Name : com.example.dao
1818 -com.example.dao > right-click > New > Interface
1819 -Name : UserDao
```

```
1820
1821     package com.example.dao;
1822
1823     import java.util.List;
1824
1825     import com.example.vo.UserVO;
1826
1827     public interface UserDao {
1828         int create(UserVO userVO);
1829         List<UserVO> readAll();
1830         UserVO read(String username);
1831         int update(UserVO userVO);
1832         int delete(String username);
1833     }
```

```
1834
1835 -com.example.dao > right-click > New > Class
1836 -Name : UserDaoImpl
```

```
1837
1838     package com.example.dao;
1839
1840     import static org.mockito.Mockito.RETURNS_DEEP_STUBS;
1841
1842     import java.sql.ResultSet;
1843     import java.sql.SQLException;
```

```
1844     import java.util.List;
1845
1846     import org.springframework.beans.factory.annotation.Autowired;
1847     import org.springframework.jdbc.core.JdbcTemplate;
1848     import org.springframework.jdbc.core.RowMapper;
1849     import org.springframework.stereotype.Repository;
1850
1851     import com.example.vo.UserVO;
1852
1853     @Repository
1854     public class UserDaoImpl implements UserDao {
1855         @Autowired
1856         private JdbcTemplate jdbcTemplate;
1857
1858         class UserMapper implements RowMapper<UserVO> {
1859             public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1860                 UserVO user = new UserVO();
1861                 user.setUsername(rs.getString("username"));
1862                 user.setAge(rs.getInt("age"));
1863                 return user;
1864             }
1865         }
1866
1867         @Override
1868         public int create(UserVO userVO) {
1869             String sql = "INSERT INTO User(username, age) VALUES(?, ?)";
1870             return this.jdbcTemplate.update(sql, userVO.getUsername(), userVO.getAge());
1871         }
1872
1873         @Override
1874         public List<UserVO> readAll() {
1875             String sql = "SELECT * FROM User";
1876             return this.jdbcTemplate.query(sql, new UserMapper());
1877         }
1878
1879         @Override
1880         public UserVO read(String username) {
1881             String sql = "SELECT * FROM User WHERE username = ?";
1882             return this.jdbcTemplate.queryForObject(sql, new Object[] {username}, new
                UserMapper());
1883         }
1884
1885         @Override
1886         public int update(UserVO userVO) {
1887             String sql = "UPDATE User SET age = ? WHERE username = ?";
1888             return this.jdbcTemplate.update(sql, userVO.getAge(), userVO.getUsername());
1889         }
1890
1891         @Override
1892         public int delete(String username) {
1893             String sql = "DELETE FROM User WHERE username = ?";
1894             return this.jdbcTemplate.update(sql, username);
1895         }
1896     }
```

```
1897     }
1898
1899 7)Controller
1900 -com.example.biz > right-click > New > Class
1901 -Name : MainController
1902
1903     package com.example.biz;
1904
1905     import org.springframework.beans.factory.annotation.Autowired;
1906     import org.springframework.stereotype.Controller;
1907     import org.springframework.ui.Model;
1908     import org.springframework.web.bind.annotation.GetMapping;
1909     import org.springframework.web.bind.annotation.PostMapping;
1910
1911     import com.example.dao.UserDao;
1912     import com.example.vo.UserVO;
1913
1914     @Controller
1915     public class MainController {
1916         @Autowired
1917         private UserDao userDao;
1918
1919         @GetMapping("/")
1920         public String index() {
1921             return "index";
1922         }
1923
1924         @PostMapping("/user")
1925         public String insert(UserVO userVO) {
1926             this.userDao.create(userVO);
1927             return "redirect:/user";
1928         }
1929
1930         @GetMapping("/user")
1931         public String list(Model model) {
1932             model.addAttribute("users", this.userDao.readAll());
1933             return "list";
1934         }
1935     }
1936
1937 8)static resources 준비
1938 -src/main/resources/static/images folder 생성
1939 --spring-boot.png 추가할 것
1940 -src/main/resources/static/js folder 생성
1941 --jquery-3.3.1.slim.min.js 추가할 것
1942 -src/main/resources/static/css folder 생성
1943 --style.css
1944
1945     @charset "UTF-8";
1946     body {
1947         background-color:yellow;
1948     }
1949     h1{
1950         color : blue;
```

```

1951     }
1952
1953 9)JSP를 위한 folder 준비
1954 -기본적으로 Spring Boot 에서는 jsp파일을 인식하지 않는다.
1955 -그래서 만일 jar로 packaging 한다면 embedded tomcat이 인식하는 web루트를 생성한다.
1956 -src > main 폴더 밑에 webapp과 jsp가 위치할 폴더를 만들어준다.
1957 -src/main folder 안에 webapp folder 생성
1958 -webapp folder 안에 WEB-INF folder 생성
1959 -WEB-INF folder 안에 jsp folder 생성
1960
1961 10)만일 JSP를 template으로 사용하는 경우에는 이 WEB-INF folder 안에 template file을 준비할 필요가 있다.
1962 11)folder를 다 만들었으면, WEB 루트(Context Root)로 인식할 있도록 환경설정을 아래와 같이 추가한다.
1963 -src/main/resources/application.properties code 추가
1964
1965 spring.mvc.view.prefix : /WEB-INF/jsp/
1966 spring.mvc.view.suffix : .jsp
1967
1968 12)jsp folder 안에 jsp file 생성
1969 -index.jsp
1970
1971 <%@ page language="java" contentType="text/html; charset=UTF-8"
1972     pageEncoding="UTF-8"%>
1973 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
1974 <!DOCTYPE html>
1975 <html>
1976 <head>
1977 <meta charset="UTF-8">
1978 <title>New User Insertion Page</title>
1979 <link rel="stylesheet" type="text/css" href="/css/style.css">
1980 <c:url var="jsurl" value="/js/jquery-3.3.1.slim.min.js" />
1981 <script src="{jsurl}"></script>
1982 <script>
1983     $(document).ready(function() {
1984         alert("Hello, Spring Boot!");
1985     });
1986 </script>
1987 </head>
1988 <body>
1989
1990     
1991     <h1>New User Insertion Page</h1>
1992     <form action="/user" method="post">
1993         Name : <input type="text" name="username" /><br />
1994         Age : <input type="number" name="age" /><br />
1995         <button type="submit">Submit</button>
1996     </form>
1997 </body>
1998 </html>
1999
2000 -list.jsp
2001
2002 <%@ page language="java" contentType="text/html; charset=UTF-8"
2003     pageEncoding="UTF-8"%>
2004 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

```

```

2005     <!DOCTYPE html>
2006     <html>
2007     <head>
2008     <meta charset="UTF-8">
2009     <title>User List</title>
2010     <link rel="stylesheet" type="text/css" href="/css/style.css">
2011     </head>
2012     <body>
2013         <h1>User List</h1>
2014         <ul>
2015             <c:forEach var="user" items="${users}">
2016                 <li>${user.username}(${user.age})</li>
2017             </c:forEach>
2018         </ul>
2019     </body>
2020 </html>

```

2022 13)src/main/java/com.example.biz/BootJdbcDemoApplication.java

```

2023
2024     package com.example.biz;
2025
2026     import org.springframework.boot.SpringApplication;
2027     import org.springframework.boot.autoconfigure.SpringBootApplication;
2028     import org.springframework.context.annotation.ComponentScan;
2029
2030     @SpringBootApplication
2031     @ComponentScan("com.example")    <-- 추가 code
2032     public class BootJdbcDemoApplication {
2033
2034         public static void main(String[] args) {
2035             SpringApplication.run(BootJdbcDemoApplication.class, args);
2036         }
2037     }

```

2039 [-@SpringBootApplication](#) 은 다음의 annotation 3개를 모두 담고 있다.

```

2040     --@SpringBootApplication
2041     --@EnableAutoConfiguration
2042     --@ComponentScan

```

2043 -만약, AutoScan이 되어야 하는 component class들 - 대표적으로 @Controller, @Service, @Repository, @Component 등-의 위치가 main class가 위치한 package보다 상위 package에 있거나, 하위가 아닌 다른 package에 있는 경우, scan이 되지 않는다.

2044 -Controller class가 main의 하위 class에 있으면 상관없지만, 예를 들어, main class가 다른 package나 하위 package가 아니면 아래와 같이 해줘야 한다.

2045 -명시적으로 ComponentScan을 할 Base Package를 지정해주면 된다.

```

2046     --ex) @ComponentScan(basePackages = "com.springboot.demo")
2047

```

2048 14)project > right-click > Run As> Spring Boot App

2049 15)<http://localhost:8080>

2052 13. Spring Boot와 JPA 연동하기

2053 -Refer to <http://jdm.kr/blog/121>

2054 1)Spring Boot의 Database 처리는 기본적으로 JPA 기술을 기반으로 하고 있다.

2055 2)이 JPA를 Spring Framework에서 사용할 수 있게 한 것이 'Spring Data JPA' framework이다.

2056 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.

2057 4)즉, 이 library는 각종 library를 조합해서 간단히 database 접속을 구현하게 한 기능이다.

2058 5)Spring Boot project 생성

2059 -Package Explorer > right-click > New > Spring Starter Project

2060 -Service URL :<http://start.spring.io>

2061 -Name : JpaDemo

2062 -Type : Maven

2063 -Packaging : Jar

2064 -Java Version : 8

2065 -Language : Java

2066 -Group : com.example

2067 -Artifact : JpaDemo

2068 -Version : 0.0.1-SNAPSHOT

2069 -Description : Demo project for Spring Boot

2070 -Package : com.example.biz

2071

2072 -Next >

2073 -Spring Boot Version : 1.5.15 (SNAPSHOT)

2074 -Check SQL > JPA, H2 Finish

2075

2076 6)Entity

2077 -JPA에서 Entity라는 것은 Database에 저장하기 위해서 정의한 class이다.

2078 -일반적으로 RDBMS에서 Table 같은 것이다.

2079 -com.example.biz > right-click > New > Class

2080 -Name : MemberVO

2081

2082 package com.example.biz;

2083

2084 import javax.persistence.Column;

2085 import javax.persistence.Entity;

2086 import javax.persistence.GeneratedValue;

2087 import javax.persistence.GenerationType;

2088 import javax.persistence.Id;

2089

2090 @Entity(name="Member")

2091 public class MemberVO {

2092 @Id

2093 @GeneratedValue(strategy = GenerationType.AUTO)

2094 private long id;

2095

2096 @Column

2097 private String username;

2098

2099 @Column

2100 private int age;

2101

2102 public MemberVO() {}

2103 public MemberVO(String username, int age) {

2104 this.username = username;

2105 this.age = age;

2106 }

2107

```

2108     public long getId() {
2109         return id;
2110     }
2111     public void setId(long id) {
2112         this.id = id;
2113     }
2114     public String getUsername() {
2115         return username;
2116     }
2117     public void setUsername(String username) {
2118         this.username = username;
2119     }
2120     public int getAge() {
2121         return age;
2122     }
2123     public void setAge(int age) {
2124         this.age = age;
2125     }
2126
2127     @Override
2128     public String toString() {
2129         return "MemberVO [id=" + id + ", username=" + username + ", age=" + age + "];
2130     }
2131 }
2132

```

-여기서 주의할 점은 기본 생성자는 반드시 넣어야 한다.

7)Repository

-Entity class를 구성했다면 이번엔 Repository interface를 만들어야 한다.

-Spring Framework에서는 Entity의 기본적인 삽입, 조회, 수정, 삭제가 가능하도록 CrudRepository라는 interface가 있다.

-com.example.biz > right-click > New > Interface

-Name : MemberRepository

```

2140
2141     package com.example.biz;
2142
2143     import java.util.List;
2144
2145     import org.springframework.data.jpa.repository.Query;
2146     import org.springframework.data.repository.CrudRepository;
2147     import org.springframework.data.repository.query.Param;
2148
2149     public interface MemberRepository extends CrudRepository<MemberVO, Long> {
2150         List<MemberVO> findByUsernameAndAgeLessThan(String username, int age);
2151
2152         @Query("select t from Member t where username= :username and age < :age")
2153         List<MemberVO> findByUsernameAndAgeLessThanSQL(@Param("username") String
                username, @Param("age") int age);
2154
2155         List<MemberVO> findByUsernameAndAgeLessThanOrderByAgeDesc(String username,
                int age);
2156     }
2157

```

-위의 코드는 실제로 MemberVO Entity를 이용하기 위한 Repository class이다.

2159 -기본적인 method 외에도 추가적인 method를 지정할 수 있다.
 2160 -method 이름을 기반(Query Method)으로 해서 만들어도 되고 @Query를 이용해 기존의 SQL처럼 만들어도 된다.
 2161 -findByUsernameAndAgeLessThan method와 findByUsernameAndAgeLessThanSQL method는 같은 결과를 출력하지만 전자의 method는 method 이름을 기반으로 한 것이고 후자의 method는 @Query annotation을 기반으로 해서 만든 것이다.
 2162 -method 이름 기반으로 해서 만들면 추후에 사용할 때 method 이름만으로도 어떤 query인지 알 수 있다는 장점이 있다.
 2163 -반대로 @Query annotation으로 만든 method는 기존의 source를 converting하는 경우 유용하게 사용할 수 있다.
 2164 -@Query
 2165 --다만 @Query annotation으로 query를 만들 때에 from 절에 들어가는 table은 Entity로 지정된 class 이름이다.
 2166 -method 이름 기반 작성법
 2167 -해당 부분은 Spring 문서
 (<https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.query-creation>)를 통해 확인할 수 있다.

8)Application 작성

2169 -Entity 및 Repository가 준비 되었다면 실제로 @SpringBootApplication annotation이 있는 class에서 실제로 사용해 보자.

```
2171 package com.example.biz;
2172
2173 import java.util.List;
2174
2175 import org.springframework.beans.factory.annotation.Autowired;
2176 import org.springframework.boot.CommandLineRunner;
2177 import org.springframework.boot.SpringApplication;
2178 import org.springframework.boot.autoconfigure.SpringBootApplication;
2179
2180 @SpringBootApplication
2181 public class JpaDemoApplication implements CommandLineRunner {
2182
2183     @Autowired
2184     MemberRepository memberRepository;
2185
2186     public static void main(String[] args) {
2187         SpringApplication.run(JpaDemoApplication.class, args);
2188     }
2189
2190     @Override
2191     public void run(String... args) throws Exception {
2192         memberRepository.save(new MemberVO("a", 10));
2193         memberRepository.save(new MemberVO("b", 15));
2194         memberRepository.save(new MemberVO("c", 10));
2195         memberRepository.save(new MemberVO("a", 5));
2196
2197         Iterable<MemberVO> list1 = memberRepository.findAll();
2198
2199         System.out.println("findAll() Method.");
2200         for( MemberVO m : list1){
2201             System.out.println(m.toString());
2202         }
2203     }
```

```

2204
2205     System.out.println("findByUserNameAndAgeLessThan() Method.");
2206     List<MemberVO> list2 = memberRepository.findByUsernameAndAgeLessThan("a",
2207         10);
2208     for( MemberVO m : list2){
2209         System.out.println(m.toString());
2210     }
2211     System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
2212     List<MemberVO> list3 = memberRepository.findByUsernameAndAgeLessThanSQL("a",
2213         10);
2214     for( MemberVO m : list3){
2215         System.out.println(m.toString());
2216     }
2217     System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
2218     List<MemberVO> list4 =
2219         memberRepository.findByUsernameAndAgeLessThanOrderByAgeDesc("a", 15);
2220     for( MemberVO m : list4){
2221         System.out.println(m.toString());
2222     }
2223     memberRepository.deleteAll();
2224 }
2225 }
2226

```

9)실행

```

2227
2228
2229     findAll() Method.
2230     Member [id=1, name=a, age=10]
2231     Member [id=2, name=b, age=15]
2232     Member [id=3, name=c, age=10]
2233     Member [id=4, name=a, age=5]
2234     findByNameAndAgeLessThan() Method.
2235     Member [id=4, name=a, age=5]
2236     findByNameAndAgeLessThanSQL() Method.
2237     Member [id=4, name=a, age=5]
2238     findByNameAndAgeLessThanSQL() Method.
2239     Member [id=1, name=a, age=10]
2240     Member [id=4, name=a, age=5]
2241
2242

```

2243 14. Lab : Spring Boot JPA

```

2244     1)Spring Boot project 생성
2245     -Package Explorer > right-click > New > Spring Starter Project
2246     -Service URL :http://start.spring.io
2247     -Name : BootJpaDemo
2248     -Type : Maven
2249     -Packaging : Jar
2250     -Java Version : 8
2251     -Language : Java
2252     -Group : com.example
2253     -Artifact : BootJpaDemo
2254     -Version : 0.0.1-SNAPSHOT

```

```
2255 -Description : Demo project for Spring Boot
2256 -Package : com.example.biz
2257
2258 -Next >
2259 -Spring Boot Version : 1.5.15 (SNAPSHOT)
2260 -Select Core > DevTools, SQL > HSQLDB, JPA
2261 -Select Web > Web, Template Engines > Thymeleaf > Finish
2262
2263 -DevTools
2264 --It provides developer tools.
2265 --These tools are helpful in application development mode.
2266 --One of the features of developer tool is automatic restart of the server for any change in
    code.
2267
2268 2)Entity 작성하기
2269 -com.example.biz > right-click > New > Package
2270 -Name : com.example.biz.vo
2271 -com.example.biz.vo > right-click > New > Class
2272 -Name : User
2273
2274 package com.example.biz.vo;
2275
2276 import javax.persistence.Column;
2277 import javax.persistence.Entity;
2278 import javax.persistence.GeneratedValue;
2279 import javax.persistence.GenerationType;
2280 import javax.persistence.Id;
2281 import javax.persistence.Table;
2282
2283 @Entity
2284 @Table
2285 public class User {
2286
2287     @Id
2288     @GeneratedValue(strategy = GenerationType.AUTO)
2289     @Column
2290     private long id;
2291
2292     @Column(length = 20, nullable = false)
2293     private String username;
2294
2295     @Column(length = 100, nullable = true)
2296     private String email;
2297
2298     @Column(nullable = true)
2299     private Integer age;
2300
2301     @Column(nullable = true)
2302     private String memo;
2303
2304     public long getId() {
2305         return id;
2306     }
2307     public void setId(long id) {
```

```
2308         this.id = id;
2309     }
2310     public String getUsername() {
2311         return username;
2312     }
2313     public void setUsername(String username) {
2314         this.username = username;
2315     }
2316     public String getEmail() {
2317         return email;
2318     }
2319     public void setEmail(String email) {
2320         this.email = email;
2321     }
2322     public Integer getAge() {
2323         return age;
2324     }
2325     public void setAge(Integer age) {
2326         this.age = age;
2327     }
2328     public String getMemo() {
2329         return memo;
2330     }
2331     public void setMemo(String memo) {
2332         this.memo = memo;
2333     }
2334 }
2335
2336 3)Repository 생성하기
2337 -com.example.biz > right-click > New > Package
2338 -Name : com.example.biz.dao
2339 -com.example.biz.dao > right-click > New > Interface
2340 -Name : UserRepository > Finish
2341
2342     package com.example.biz.dao;
2343
2344     import org.springframework.data.jpa.repository.JpaRepository;
2345     import org.springframework.stereotype.Repository;
2346
2347     import com.example.biz.vo.User;
2348
2349     @Repository("repository")
2350     public interface UserRepository extends JpaRepository<User, Long>{
2351
2352     }
2353 -JpaRepository라는 interface는 새로운 repository를 생성하기 위한 토대가 된다.
2354 -모든 Repository는 이 JpaRepository를 상속해서 작성한다.
2355
2356 4)HelloController 작성
2357 -com.example.biz > right-click > New > Class
2358 -Name : HelloController
2359
2360     package com.example.biz;
2361
```

```
2362 import org.springframework.beans.factory.annotation.Autowired;
2363 import org.springframework.stereotype.Controller;
2364 import org.springframework.web.bind.annotation.RequestMapping;
2365 import org.springframework.web.servlet.ModelAndView;
2366
2367 import com.example.biz.dao.UserRepository;
2368 import com.example.biz.vo.User;
2369
2370 @Controller
2371 public class HelloController {
2372
2373     @Autowired
2374     UserRepository repository;
2375
2376     @RequestMapping("/")
2377     public ModelAndView index(ModelAndView mav) {
2378         mav.setViewName("index");
2379         mav.addObject("msg", "this is sample content.");
2380         Iterable<User> list = repository.findAll();
2381         mav.addObject("data", list);
2382         return mav;
2383     }
2384 }
```

- 2386 -UserRepository에는 findAll 같은 method가 정의되어 있지 않다.
- 2387 -이것은 부모 interface인 JpaRepository가 가지고 있는 method이다.
- 2388 -이를 통해 모든 entity가 자동으로 추출되는 것이다.

2390 5)JUnit Test

- 2391 -src/test/java/com.example.biz.BootJpaDemoApplicationTests.java > right-click > Run As > JUnit Test
- 2392 -Green bar

2394 6)template 준비하기

- 2395 -src/main/resources > right-click > New > File
- 2396 -File name : messages.properties
- 2397
- 2398 content.title=Message sample page.
- 2399 content.message=This is sample message from properties.
- 2400
- 2401 -src/main/resources/static > right-click > New > Web > HTML Files
- 2402 -Name : index.html

```
2403
2404 <!DOCTYPE HTML>
2405 <html xmlns:th="http://www.thymeleaf.org">
2406 <head>
2407     <title>top page</title>
2408     <meta charset="UTF-8" />
2409     <style>
2410         h1 { font-size:18pt; font-weight:bold; color:gray; }
2411         body { font-size:13pt; color:gray; margin:5px 25px; }
2412         pre { border: solid 3px #ddd; padding: 10px; }
2413     </style>
2414 </head>
```

```

2415     <body>
2416         <h1 th:text="#{content.title}">Hello page</h1>
2417         <pre th:text="${data}"></pre>
2418     </body>
2419 </html>
2420
2421 -src/main/resources > right-click > New > File
2422 --File name : messages.properties > Finish
2423
2424     content.title=Message sample page.
2425     content.message=This is sample message from properties.
2426
2427 -실행해서 접속
2428 --저장돼있는 data가 회색 사각 틀 안에 표시된다.
2429 -아직 아무 data가 없기 때문에 빈 배열 []라고 표시된다.
2430
2431 7)Entity의 CRUD 처리하기 : form으로 data 저장하기
2432 -index.html 수정
2433
2434     <!DOCTYPE HTML>
2435     <html xmlns:th="http://www.thymeleaf.org">
2436     <head>
2437         <title>top page</title>
2438         <meta charset="UTF-8" />
2439         <style>
2440             h1 { font-size:18pt; font-weight:bold; color:gray; }
2441             body { font-size:13pt; color:gray; margin:5px 25px; }
2442             tr { margin:5px; }
2443             th { padding:5px; color:white; background:darkgray; }
2444             td { padding:5px; color:black; background:#e0e0ff; }
2445         </style>
2446     </head>
2447     <body>
2448         <h1 th:text="#{content.title}">Hello page</h1>
2449         <table>
2450             <form method="post" action="/" th:object="${formModel}">
2451                 <tr><td><label for="username">이름</label></td>
2452                 <td><input type="text" name="username" th:value="*{username}" /></td></tr>
2453                 <tr><td><label for="age">연령</label></td>
2454                 <td><input type="text" name="age" th:value="*{age}" /></td></tr>
2455                 <tr><td><label for="email">메일</label></td>
2456                 <td><input type="text" name="email" th:value="*{email}" /></td></tr>
2457                 <tr><td><label for="memo">메모</label></td>
2458                 <td><textarea name="memo" th:text="*{memo}"
2459                     cols="20" rows="5"></textarea></td></tr>
2460                 <tr><td></td><td><input type="submit" /></td></tr>
2461             </form>
2462         </table>
2463         <hr/>
2464         <table>
2465             <tr><th>ID</th><th>이름</th></tr>
2466             <tr th:each="obj : ${datalist}">
2467                 <td th:text="${obj.id}"></td>
2468                 <td th:text="${obj.username}"></td>

```



```

2469         </tr>
2470     </table>
2471 </body>
2472 </html>
2473
2474 -HelloController.java 수정
2475
2476     package com.example.biz;
2477
2478     import org.springframework.beans.factory.annotation.Autowired;
2479     import org.springframework.stereotype.Controller;
2480     import org.springframework.transaction.annotation.Transactional;
2481     import org.springframework.web.bind.annotation.ModelAttribute;
2482     import org.springframework.web.bind.annotation.RequestMapping;
2483     import org.springframework.web.bind.annotation.RequestMethod;
2484     import org.springframework.web.servlet.ModelAndView;
2485
2486     import com.example.biz.dao.UserRepository;
2487     import com.example.biz.vo.User;
2488
2489     @Controller
2490     public class HelloController {
2491
2492         @Autowired
2493         UserRepository repository;
2494
2495         @RequestMapping(value = "/", method = RequestMethod.GET)
2496         public ModelAndView index(@ModelAttribute("formModel") User mydata, ModelAndView
2497             mav) {
2498             mav.setViewName("index");
2499             mav.addObject("msg","this is sample content.");
2500             Iterable<User> list = repository.findAll();
2501             mav.addObject("datalist",list);
2502             return mav;
2503         }
2504
2505         @RequestMapping(value = "/", method = RequestMethod.POST)
2506         @Transactional(readOnly=false)
2507         public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
2508             mav) {
2509             repository.saveAndFlush(mydata);
2510             return new ModelAndView("redirect:/");
2511         }
2512     }
2513
2514 -접속해서 실행
2515     --form에 text를 입력해서 전송하면 data가 추가돼서 아래쪽 table에 표시된다.
2516
2517 8)@ModelAttribute와 data 저장
2518     -@ModelAttribute
2519     --이것은 entity class의 instance를 자동으로 적용할 때 사용
2520     --인수에는 instance 이름을 지정한다.
2521     --이것은 전송 form에서 th:object로 지정하는 값이 된다.
2522     --전송된 form의 값이 자동으로 User instance로 저장된다.

```

2521 --따라서 이 annotation을 이용하면 이렇게 쉽게 전송한 data를 저장할 수 있다.

2522

2523 9)saveAndFlush() method

2524 -HomeController.java의 아래 code를 보자.

2525

2526 @RequestMapping(value = "/", method = RequestMethod.POST)

2527 @Transactional(readOnly=false)

2528 public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView mav) {

2529 repository.saveAndFlush(mydata);

2530 return new ModelAndView("redirect:/");

2531 }

2532

2533 -미리 설정한 entity는 JpaRepository의 saveAndFlush라는 method를 통해 entity를 영구화한다.

2534 -Database를 사용하고 있다면 Database에 그대로 저장된다.

2535

2536 10)@Transactional과 transaction

2537 -바로 위의 code에서 @Transactional(readOnly=false)가 있다.

2538 -이 annotation은 transaction을 위한 것이다.

2539 -이 annotation때문에 method내에서 실행되는 database 처리가 일괄적으로 실행되게 된다.

2540 -data 변경 처리는 도중에 외부 접속에 의해 data 구조나 내용이 바뀌면 data 일관성에 문제가 발생하게 된다.

2541 -이런 문제를 방지하기 위해 transaction이 사용되는 것이다.

2542 -code를 보면 readOnly=false라고 설정하고 있다.

2543 -이 readOnly는 문자 그대로 '읽기 전용(변경 불가)'임을 의미한다.

2544 -readOnly=false라고 설정하면 변경을 허가하는 transaction이다.

2545

2546 11)Data 초기화 처리

2547 -저장한 data는 application으로 종료하고 다시 실행하면 지워진다.

2548 -HSQLDB는 기본적으로 memory내에 data를 cache하고 있으므로 종료와 함께 지워지는 것이다.

2549 -controller에 data를 작성하는 초기화 처리를 넣기로 한다.

2550 -HelloController.java code 추가

2551

2552 @PostConstruct

2553 public void init(){

2554 User user1 = new User();

2555 user1.setUsername("한지민");

2556 user1.setAge(24);

2557 user1.setEmail("javaexpert@nate.com");

2558 user1.setMemo("Hello, Spring JPA");

2559 repository.saveAndFlush(user1);

2560

2561 User user2 = new User();

2562 user2.setUsername("조용필");

2563 user2.setAge(66);

2564 user2.setEmail("aaa@aaa.com");

2565 user2.setMemo("Good Morning!");

2566 repository.saveAndFlush(user2);

2567

2568 User user3 = new User();

2569 user3.setUsername("이미자");

2570 user3.setAge(70);

2571 user3.setEmail("bbb@bbb.com");

2572 user3.setMemo("Spring Boot is very good.");

2573 repository.saveAndFlush(user3);

```

2574     }
2575
2576 -@PostConstruct는 생성자를 통해 instance가 생성된 후에 호출되는 method임을 나타낸다.
2577 -Controller는 처음에 한 번만 instance를 만들고 이후에는 해당 instance를 유지한다.
2578 -따라서 여기에 test용 data 작성 처리를 해두면 application 실행시에 반드시 한 번 실행되어, data가 준비되는 것
    이다.
2579
2580 12)User Find 및 Update 처리하기
2581 -src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
2582 -File name : edit.html > Finish
2583
2584 <!DOCTYPE HTML>
2585 <html xmlns:th="http://www.thymeleaf.org">
2586 <head>
2587     <title>edit page</title>
2588     <meta charset="UTF-8" />
2589     <style>
2590         h1 { font-size:18pt; font-weight:bold; color:gray; }
2591         body { font-size:13pt; color:gray; margin:5px 25px; }
2592         tr { margin:5px; }
2593         th { padding:5px; color:white; background:darkgray; }
2594         td { padding:5px; color:black; background:#e0e0ff; }
2595     </style>
2596 </head>
2597 <body>
2598     <h1 th:text="${title}">Edit page</h1>
2599     <table>
2600     <form method="post" action="/edit" th:object="${formModel}">
2601         <input type="hidden" name="id" th:value="*{id}" />
2602         <tr><td><label for="username">이름</label></td>
2603             <td><input type="text" name="username" th:value="*{username}" /></td></tr>
2604         <tr><td><label for="age">연령</label></td>
2605             <td><input type="text" name="age" th:value="*{age}" /></td></tr>
2606         <tr><td><label for="email">메일</label></td>
2607             <td><input type="text" name="email" th:value="*{email}" /></td></tr>
2608         <tr><td><label for="memo">메모</label></td>
2609             <td><textarea name="memo" th:text="*{memo}"
2610                 cols="20" rows="5"></textarea></td></tr>
2611         <tr><td></td><td><input type="submit" /></td></tr>
2612     </form>
2613     </table>
2614 </body>
2615 </html>
2616
2617 -UserRepository code 추가
2618
2619 @Repository("repository")
2620 public interface UserRepository extends JpaRepository<User, Long> {
2621     public User findById(Long id);
2622 }
2623
2624 -RequestHandler 작성하기
2625 ---HelloController.java code 추가
2626

```

```

2627     @RequestMapping(value = "/edit/{id}", method = RequestMethod.GET)
2628     public ModelAndView edit(@ModelAttribute User user, @PathVariable int id,
2629                             ModelAndView mav) {
2630         mav.setViewName("edit");
2631         mav.addObject("title","edit mydata.");
2632         User findUser = repository.findById((long)id);
2633         mav.addObject("formModel", findUser);
2634         return mav;
2635     }
2636
2637     @RequestMapping(value = "/edit", method = RequestMethod.POST)
2638     @Transactional(readOnly=false)
2639     public ModelAndView update(@ModelAttribute User user, ModelAndView mav) {
2640         repository.saveAndFlush(user);
2641         return new ModelAndView("redirect:/");
2642     }

```

-접속해서 아래와 같이 URL을 입력하면

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

-해당 ID의 data가 표시된다.

-data를 변경하고 전송해보자.

-findById는 어디서 구현되는 것일까?

--repository는 method의 이름을 기준으로 entity 검색 처리를 자동 생성한다.

--즉 repository에 method 선언만 작성하고, 구체적인 처리를 구현할 필요가 없다.

13)Entity delete 구현하기

-update를 하고 select를 했으니 이번에는 delete를 해 보자.

-delete.html template를 작성한다.

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

-File name : delete.html > Finish

```

2659 <!DOCTYPE HTML>
2660 <html xmlns:th="http://www.thymeleaf.org">
2661 <head>
2662     <title>edit page</title>
2663     <meta charset="UTF-8" />
2664     <style>
2665         h1 { font-size:18pt; font-weight:bold; color:gray; }
2666         body { font-size:13pt; color:gray; margin:5px 25px; }
2667         td { padding:0px 20px; background:#eee;}
2668     </style>
2669 </head>
2670 <body>
2671     <h1 th:text="${title}">Delete page</h1>
2672     <table>
2673     <form method="post" action="/delete" th:object="${formModel}">
2674         <input type="hidden" name="id" th:value="*{id}" />
2675         <tr><td><p th:text="|이름 :   *{username}|"></p></td></tr>
2676         <tr><td><p th:text="|연령 :   *{age}|" ></p></td></tr>
2677         <tr><td><p th:text="*{email}"></p></td></tr>
2678         <tr><td><p th:text="*{memo}"></p></td></tr>
2679         <tr><td><input type="submit" value="delete"/></td></tr>

```

```
2680         </form>
2681     </table>
2682 </body>
2683 </html>
2684
2685 -RequestHandler 작성
2686
2687     @RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
2688     public ModelAndView delete(@PathVariable int id, ModelAndView mav) {
2689         mav.setViewName("delete");
2690         mav.addObject("title","delete mydata.");
2691         User user = repository.findById((long)id);
2692         mav.addObject("formModel",user);
2693         return mav;
2694     }
2695
2696     @RequestMapping(value = "/delete", method = RequestMethod.POST)
2697     @Transactional(readOnly=false)
2698     public ModelAndView remove(@RequestParam long id, ModelAndView mav) {
2699         repository.delete(id);
2700         return new ModelAndView("redirect:/");
2701     }
2702 -접속해서 실행
2703     http://localhost:8080/delete/2라고 하면 id가 2번이 출력되고 delete button을 누르면 삭제된다.
```