```
HOL : Spring Boot
----------------------------------------------------------------
Task1. Maven으로 Spring Boot Project 생성하기
1. In Package Explorer
  1)right-click > New > Project > Maven > Maven Project > Next
  2)[New Maven project] 창에서 > Next

  3)Select an Archetype
    -Group Id : org.apache.maven.archetypes
    -Artifact Id : maven-archetype-quickstart
    -Version : 1.4
    -Next

  4)Enter an artifact id
    -Group Id : com.example
    -Artifact Id : springbootdemo
    -Version : 0.0.1-SNAPSHOT
    -Package : com.example.springbootdemo
    -Finish


2. pom.xml 수정
  <?xml version="1.0" encoding="UTF-8"?>

  <project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

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

    <name>springbootdemo</name>
    <!-- FIXME change it to the project's website -->
    <url>http://www.example.com</url>
    <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>2.2.6.RELEASE</version>
    </parent>

    <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <maven.compiler.source>13</maven.compiler.source>
      <maven.compiler.target>13</maven.compiler.target>
    </properties>

    <dependencies>
      <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13</version>
        <scope>test</scope>
      </dependency>
      <dependency>
        <groupId>org.springframework.boot</groupId>
```

```
        <artifactId>spring-boot-starter-web</artifactId>
      </dependency>
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
      </dependency>
    </dependencies>
    ...
    ...
```

1)<parent> 설정하기
-Spring Boot의 설정 정보를 상속한다.
-여기서 지정한 version이 spring boot의 version이 된다.
-spring boot의 version을 올리려면 <version> tag 안에 있는 설정 값을 변경한다.

2)spring-boot-starter-web
-spring boot로 web application을 만들 때 참조할 기본 library 정보를 설정한다.
-이렇게 쓰기만 해도 web application 제작에 필요한 spring framework 관련 library와 third-party library를 이용할 수 있게 된다.
-version은 위 parent에서 설정한 spring-boot-starter-parent 안에 정의되어 있으므로, 여기서는 지정하지 않아도 된다.

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

3. Project > right-click > Properties
1)Java Build Path > Modulepath > JRE System Library [jdk-13.0.2] > Apply
2)Java Compiler > JDK Compliance > 13 > Apply
3)Project Facets > Java 13 > Runtimes tab > Check [jdk-13.0.2]
4)Apply and Close

4. Project > right-click > Maven > Update Project... > OK

5. Hello World!를 출력하는 Web application 작성하기
1)src/main/java/com/example/springbootdemo/App.java

```
  package com.example.springbootdemo;

  import org.springframework.boot.SpringApplication;
  import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
  import org.springframework.web.bind.annotation.RequestMapping;
  import org.springframework.web.bind.annotation.RestController;

  /**
  * Hello world!
  *
  */
  @RestController
  @EnableAutoConfiguration
  public class App {
    @RequestMapping("/")
    String home(){
      return "Hello, World!";
    }
```

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

2. [New Spring Starter Project Dependencies] 창에서
   1)Spring Boot Version : 2.2.6
   2)Select Web > check Spring Web > Finish

3. src/main/java/com.example.demo.DemoApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

  public static void main(String[] args) {
    SpringApplication.run(DemoApplication.class, args);
  }
}
```

4. DemoApplication.java > right-click > Run As > Spring Boot App

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

2020-04-20 22:03:17.462  INFO 15496 --- [           main] com.example.demo.DemoApplication         : Starting DemoApplication on DESKTOP-1BKHISM with PID 15496 (C:\SpringHome\demo\target\classes started by devex in C:\SpringHome\demo)
2020-04-20 22:03:17.464  INFO 15496 --- [           main] com.example.demo.DemoApplication         : No active profile set, falling back to default profiles: default
2020-04-20 22:03:18.048  INFO 15496 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2020-04-20 22:03:18.055  INFO 15496 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-04-20 22:03:18.055  INFO 15496 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.33]
2020-04-20 22:03:18.120  INFO 15496 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2020-04-20 22:03:18.120  INFO 15496 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 626 ms
2020-04-20 22:03:18.246  INFO 15496 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2020-04-20 22:03:20.066  INFO 15496 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2020-04-20 22:03:20.069  INFO 15496 --- [           main] com.example.demo.DemoApplication         : Started DemoApplication in 2.811 seconds
```

(JVM running for 5.007)

5. http://localhost:8080
Whitelabel Error Page
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Mon Apr 20 22:04:04 KST 2020
There was an unexpected error (type=Not Found, status=404).
No message available

6. src/main/java/com.example.demo.DemoApplication.java 수정하기

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@EnableAutoConfiguration
public class DemoApplication {

  @RequestMapping("/")
  String home() {
    return "Hello, World!";
  }

  public static void main(String[] args) {
    SpringApplication.run(DemoApplication.class, args);
  }

}
```

1)빨간색 실행 중비 button click
2)DemoApplication.java > right-click > Run As > Spring Boot App
3)http://localhost:8080/
Hello, World!

Task3. Groovy로 Application 개발하기

1. 준비
1)Visit
https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html

2)3.2.1 Manual Installation에서 spring-boot-cli-2.2.4.RELEASE-bin.zip link를 click한다.
3)Unzip > Move to C:\Program Files\spring-2.2.4.RELEASE
4)path 설정
-%PATH%;C:\Program Files\spring-2.2.1.RELEASE\bin

2. Groovy Script 작성하기
1)Editor(예:VSCode)를 열어서 아래의 code를 적당한 위치(즉 C:\temp)에 file 이름은 app.groovy라고 저

장한다.

2)app.groovy

```
@RestController
class App {

  @RequestMapping("/")
  def home() {
    "Hello!!!"
  }
}
```

3)app.groovy 실행하기

-Command Prompt에서,

```
$ cd C:\temp
$ spring run app.groovy
Resolving dependencies.....................................

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.4.RELEASE)

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)
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
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)
2020-02-18 23:22:22.531  INFO 12048 --- [      runner-0]
o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-02-18 23:22:22.532  INFO 12048 --- [      runner-0]
org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache
Tomcat/9.0.30]
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].
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].
2020-02-18 23:22:22.541  INFO 12048 --- [      runner-0]
o.a.catalina.core.AprLifecycleListener   : APR/OpenSSL configuration: useAprConnector
[false], useOpenSSL [true]
2020-02-18 23:22:22.552  INFO 12048 --- [      runner-0]
o.a.catalina.core.AprLifecycleListener   : OpenSSL successfully initialized [OpenSSL 1.1.1c
 28 May 2019]
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$DefaultScopeParentC
lassLoader@6e84d437] of class [class
org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentCl
assLoader]
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>
```

```
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <title>Index Page</title>
  <style type="text/css">
    h1 {
      font-size: 18pt;
      font-weight: bold;
      color: gray;
    }

    body {
      font-size: 13pt;
      color: gray;
      margin: 5px 25px;
    }
  </style>
</head>

<body>
  <h1>Hello! Spring Boot with Thymeleaf</h1>
  <p>This is sample web page.</p>
</body>

</html>
```

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

-app.groovy

```
@Grab("thymeleaf-spring5")

@Controller
class App {

  @RequestMapping("/")
  @ResponseBody
  def home(ModelAndView mav) {
    mav.setViewName("home")
    mav
  }
}
```

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

```
<!doctype html>
<html lang="en">
```

```
      <head>
        <meta charset="UTF-8" />
        <title>Index Page</title>
        <style type="text/css">
          h1 { font-size:18pt; font-weight:bold; color:gray; }
          body { font-size:13pt; color:gray; margin:5px 25px; }
        </style>
      </head>
      <body>
        <h1>Hello!</h1>
        <p th:text="${msg}">${msg}</p>
        <form method="post" action="/send">
          <input type="text" name="text1" th:value="${value}" />
          <input type="submit" value="Send" />
        </form>
      </body>
    </html>

  2)app.groovy
    @Grab("thymeleaf-spring5")
    @Controller
    class App {

      @RequestMapping(value = "/", method=RequestMethod.GET)
      @ResponseBody
      def home(ModelAndView mav) {
        mav.setViewName("home")
        mav.addObject("msg", "Please write your name...")
        mav
      }
      @RequestMapping(value = "/send", method=RequestMethod.POST)
      @ResponseBody
      def send(@RequestParam("text1") String str, ModelAndView mav){
        mav.setViewName("home")
        mav.addObject("msg", "Hello, " + str + "!!!")
        mav.addObject("value", str)
        mav
      }
    }

  3)script 실행
    $ spring run app.groovy

  4)http://localhost:8080
    Hello!
    Please write your name...

    Hello, 한지민!!!


------------------------------------------------------------
Task4. SPRING INITIALIZR(Maven)
1. Visit http://start.spring.io/
2. 설정
  1)Maven Project
  2)Java
  3)2.2.6
  4)Group : com.example
```

5)Artifact : demo
6)Name : demo
7)Description : Demo project for Spring Boot
8)Package Name : com.example.demo
9)Packaging : Jar
10)Java Version : 8
11)Dependencies : [ADD DEPENDENCIES] click > Spring Web

12)Click [Generate]
13)Downloads [demo.zip] : 55.5KB
14)Unpack to Spring workspace.

3. Project Import
1)In Package Explorer > right-click > Import > Maven > Existing Maven Projects > Next
2)Click [Browse...] > demo Folder Select > Finish

4. JUnit Test
1)src/test/java/com.example.demo.DemoApplicationTests.java > right-click > Run As > JUnit Test >
Green bar

5. Spring Boot App 실행하기
1)demp project > right-click > Run As > Spring Boot App

2)http://localhost:8080/

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.
Thu Nov 07 15:47:32 KST 2019
There was an unexpected error (type=Not Found, status=404).
No message available

6. Controller 생성
1)src/main/java/com.example.demo > right-click > New > Class
2)Name : HelloController
3)Finish

```
package com.example.demo;

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

@RestController
public class HelloController {

  @GetMapping("/")
  public String hello() {
    return "Hello, Spring Boot World";
  }
}
```

7. Relaunch demo

-http://localhost:8080/

Hello, Spring Boot World

8. RestController 사용하기
1)HelloController.java 수정하기

```
@RestController
public class HelloController {

  @GetMapping("/hello")
  public String hello(String name) {
    return "Hello! : " + name;
  }
}
```

2)@RestController
-이 annotation을 붙이면 web appication에서 request을 받아들이는 controller class임을 나타낸다.
-Spring 4부터 지원
-REST 방식의 응답을 처리하는 Controller를 구현할 수 있다.
-@Controller를 사용할 때 method의 return type이 문자열일 경우, 문자열에 해당하는 View를 만들어야 하지만, Controller를 RestController를 사용할 경우에는 Return되는 문자열이 Browser에 그대로 출력되기 때문에 별도로 View 화면을 만들 필요가 없다.

3)Relaunch demo
-http://localhost:8080/hello?name=한지민
Hello : 한지민

9. VO 사용하기
1)pom.xml 수정
-lombok library 추가하기

```
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
   <groupId>org.projectlombok</groupId>
   <artifactId>lombok</artifactId>
   <version>1.18.12</version>
   <scope>provided</scope>
</dependency>
```

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

2)com.example.demo/DemoApplication.java 수정하기

```
@SpringBootApplication
@ComponentScan(basePackages = {"com.example"})
public class DemoApplication {
  ...
}
```

2)com.example.vo package 생성
3)com.example.vo.UserVO Class 생성

```
package com.example.vo;

import lombok.Data;
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>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
      </dependency>
```

11)방금 추가된 DevTools 를 적용하기 위해 Application을 다시 실행한다.
12)Controller의 Code를 수정하면 자동으로 Restart가 일어난다.
13)Browser에서 Refresh를 누르면 수정된 Code가 반영된다.
14)즉, Java Code를 수정한 뒤 Application을 다시 수동으로 시작하지 않아도 된다

12. Lombok Library 사용하기

1)보통 VO Class를 사용할 때 Table의 Column 이름과 같은 이름을 사용한다.
2)getter / setter method도 생성하고 toString() method도 생성한다.
3)하지만 code가 지저분해지고 모든 VO class와 JPA에서 사용할 Domain Class에 이런 Method를 반복적으로 작성하는 일은 사실 번거로운 일이다.
4)이런 문제를 간단하게 해결하기 위한 Library가 Lombok이다.
5)Lombok을 사용하면 Java File을 Compile할 때, 자동으로 생성자, getter / setter, toString() 같은 code들을 추가해준다.
6)현재 사용하고 있는 project에 Lombok Library를 추가해 보자.
7)위처럼 pom.xml의 <dependency> tag 제일 마지막에 Ctrl + Space 단축키를 누른다.
8)Context Menu에서 [Edit Starters...]를 double-click한다.
9)[Pom file needs savings] 창에서 pom.xml 파일의 수정 내용을 저장할 것인지 물어보면 [Save Pom] button을 클릭한다.
10)[Edit Spring Boot Starters]창에서 Developer Tools > Lombok 체크한다.
11)그리고 [OK] 클릭한다.
12)그러면 pom.xml에 다음과 같은 Code가 추가된다.

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

13)Lombok을 사용하려면 별도로 STS 설치 Folder에 Lombok Library를 추가해야 한다.
14)STS에 Lombok Library를 추가하기 위해 STS를 일단 종료한다.
15)Lombok Homepage(https://projectlombok.org/)를 방문한다.
16)download page로 이동하여 현재 최신 버전인 1.18.10을 Downloads 한다.
17)download 한 Folder로 이동하여 Cmd 창에서 아래의 명령을 수행한다.

```
    java -jar lombok.jar
```

18)[Project Lombok v1.18.10 - Installer] 창에서, IDEs에 보면 현재 Eclipse와 STS가 설치된 folder가 자동감지된다.
19)확인이 되었으면 [Install/Update] button click한다.
20)[Quit Installer] button click 한다.
21)Lombok이 설치되면 STS 설치 Folder(C:\Program Files\sts-4.4.0.RELEASE)에 lombok.jar가 있는 것을 확인할 수 있다.
22)다시 STS를 실행하여 UserVO.java로 들어간다.

```
    package com.example.vo;

    import lombok.Getter;
    import lombok.Setter;
    import lombok.ToString;

    @Getter
    @Setter
    @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, @EquqlsAndHashCode 모두
        생성.
739
740
741
742 ------------------------------------------------------
743 Task5. SPRING INITIALIZR(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       }
```

```
    }

  3)Relaunch demo
  4)http://localhost:8080/
    Hello, Spring Boot World



----------------------------------------------------------
Task6. 사용자 정의 Starter 만들기
1. Maven Project 생성
  1)Project Explorer > right-click > New > Maven Project
  2)Next
  3)org.apache.maven.archetypes, maven-archetype-quickstart, 1.4 > Next
  4)Group Id : com.example
    -Artifact Id : mybootstarter
    -Version : 0.0.1-SNAPSHOT
    -Package : com.example.mybootstarter
    -Finish


2. Project Facets 수정하기
  1)mybootstarter Project > right-click > Properties > Project Facets
  2)Java 1.8 > Runtimes Tab > Apache Tomcat v9.0, jdk1.8.0_241 check
  3)Apply and Close click


3. Mvnrepository에서 'spring boot'로 검색
  1)Spring Boot Autoconfigure > 2.2.4.RELEASE
  2)아래 코드 복사 후 pom.xml 에 붙여넣기

    <!--
    https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-autoconfigure
    -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-autoconfigure</artifactId>
      <version>2.2.4.RELEASE</version>
    </dependency>

  3)pom.xml > right-click > Run As > Maven install
  [INFO] BUILD SUCCESS


4. dependencyManagement 추가
  1)앞으로 추가되는 Library들의 Version을 일괄적으로 관리하기 위해 pom.xml에 Code 추가
  2)Mvnrepository에서 'spring boot dependencies'로 검색
  3)Spring Boot Dependencies에서 3.0.1.RELEASE
  4)아래의 Code를 pom.xml의 제일 아래에 추가

    <!--
    https://mvnrepository.com/artifact/org.apache.camel.springboot/camel-spring-boot-dependencies -->
    <dependency>
      <groupId>org.apache.camel.springboot</groupId>
      <artifactId>camel-spring-boot-dependencies</artifactId>
      <version>3.0.1</version>
      <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-dependencies -->
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) {
```

```
            e.printStackTrace();
          }
          return null;
        }
      }
```

4)JdbcConnectionManager를 bean으로 등록하는 환경 설정 Class를 작성한다.
5)com.example.config package 생성
6)com.example.config.UserAutoConfiguration.java 생성

```
package com.example.config;

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

import com.example.util.JdbcConnectionManager;

@Configuration
public class UserAutoConfiguration {
  @Bean
  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;
  }
}
```

6. src/main/resources folder 추가
1)mybootstarter project > right-click > New > Source Folder
2)Folder name : src/main/resources
3)Finish

7. resources/META-INF folder 생성
1)src/main/resources > right-click > New > Folder
2)Folder name : META-INF
3)Finish

8. src/main/resources/META-INF/spring.factories file 생성
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\com.example.config.UserAutoConfiguration

9. Build
1)mybootstarter project > right-click > Run As > Maven install
[INFO] BUILD SUCCESS

2)성공하면 C:\Users\계정\.m2\repository\com\example\mybootstarter\0.0.1-SNAPSHOT에 packaging한 jar 파일이 등록되어 있을 것이다.

10. Starter와 자동 설정 사용하기
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 @Service를 추가했다.
10)ApplicationRunner Interface를 구현했기 때문에 JdbcConnectionManagerRunner 객체가 생성되자마자 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,
```

url=jdbc:oracle:thin:@localhost:1521:XE, username=hr, password=hr]

13)만일 자동설정이 동작하지 않았다면 의존성 주입에서 Error가 발생했을 것이다

11. 자동설정 재정의하기

1)Bean 재정의하기

-현재 mybootstarter는 Oracle과 Connection을 할 수 있다.

-이것을 MariaDB로 변경하려고 한다.

-demo Project에 com.example.config package 생성한다.

-com.example.config.UserConfiguration Class를 생성한다.

```
package com.example.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import com.example.util.JdbcConnectionManager;

@Configuration
public class UserConfiguration {

  @Bean
  public JdbcConnectionManager getJdbcConnectionManager() {
    JdbcConnectionManager manager = new JdbcConnectionManager();
    manager.setDriverClass("org.mariadb.jdbc.Driver");
    manager.setUrl("jdbc:mariadb://localhost:3306/test");
    manager.setUsername("root");
    manager.setPassword("javamariadb");
    return manager;
  }
}
```

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

-그런데, Application을 실행해 보면 Console에는 Error가 발생한다.

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

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

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

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

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

※Properties Editor Plugin 설치하기

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

Open With > PropertiesEditor
10. 한글이 깨지지 않고 정상적으로 보이는 것을 볼 수 있다.

-demo Project를 다시 실행하면 Error는 나오지 않는데, 아직도 Oracle의 설정 정보가 나오는 것을 볼 수 있다.
-그 이유는 demo Project의 bean으로 등록한 JdbcConnectionManager가 사용된 것이 아니라 mybootstarter Project에서 등록한 JdbcConnectionManager를 사용했기 때문이다.
-이 문제를 해결하기 위한 Annotation이 바로 @Conditional이다.
-@Conditional Annotation은 조건에 따라 새로운 객체를 생성할지 안할지를 결정할 수 있다.

2)@Conditional Annotation 사용하기
-@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 Connecitor)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 @Configuration과

```
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.properites 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-processotr 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
```

[INFO] BUILD SUCCESS
-demo Project를 Refresh하고 다시 실행한다.
--다시 Oracle 설정 정보가 나오는 것을 알 수 있다.
Connection Manager : JdbcConnectionManager
[driverClass=oracle.jdbc.driver.OracleDriver,
url=jdbc:oracle:thin:@localhost:1521:XE, username=hr, password=hr]
-만일 Database가 다시 MaraiDB로 변경된다면 demo Project의 application.properties를 다음과 같이 변경하면 된다.
Bean Overriding 설정
spring.main.allow-bean-definition-overriding=true
데이터 소스 : Oracle
#user.jdbc.driverClass=oracle.jdbc.driver.OracleDriver
#user.jdbc.url=jdbc:oracle:thin:@localhost:1521:XE
#user.jdbc.username=hr
#user.jdbc.password=hr
데이터 소스 : MariaDB
user.jdbc.driverClass=org.mariadb.jdbc.Driver
user.jdbc.url=jdbc:mariadb://localhost:3306/test
user.jdbc.username=root
user.jdbc.password=javamariadb

-저장하면 바로 MaraiDB로 설정정보가 변경된 것을 알 수 있다.
Connection Manager : JdbcConnectionManager [driverClass=org.mariadb.jdbc.Driver,
url=jdbc:mariadb://localhost:3306/test, username=root, password=javamariadb]

Task7. 간단한 JPA Project
1. H2 Database 설치하기
1)여러 RDBMS가 있지만 H2를 사용하려는 이유는 Spring Boot가 기본적으로 H2를 지원하고 있기 때문이다.
2)H2는 Java로 만들어졌으며, 용량이 작고 실행 속도가 빠른 Open Source Database이다.
3)H2 Homepage(http://www.h2database.com/html/main.html)를 방문한다.
4)Main page에서 Download의 All Platforms (zip, 8MB) Link를 Click하여 압축파일을 Download한다.
5)h2-2019-10-14.zip 압축을 풀고 h2 Folder를 C:/Program Files로 이동한다.
6)h2/bin의 h2w.bat를 실행하면 Browser기반의 관리 Console이 열린다.

7)Tray에 있는 H2 Database Engine > right-click > Create a Database...을 실행하여 다음의 각 항목에 값을 입력하고 [Create] button을 click한다.
-Database path : ./test
-Username : sa
-Password : javah2
-Password confirmation : javah2
-Create button click

Database was created successfully.
JDBC URL for H2 Console:
jdbc:h2:./test

8)각 항목의 정보를 입력하고 [Test Connection] 클릭해본다.
--Driver Class : org.h2.Driver
--JDBC URL : jdbc:h2:~/test
--User Name : sa
--Password : javah2

9)연결이 성공하면 Web Console이 열린다.

2. JPA Project Installation
  1)In STS, Help > Install New Software
  2)Work with : https://download.eclipse.org/releases/2019-12
  3)Filter : jpa
  4)결과에서
    Web, XML, Java EE and OSGi Enterprise Development 하위의
    -Dali Java Persistence Tools - EclipseLink JPA Support
    -Dali Java Persistence Tools - JPA Diagram Editor
    -Dali Java Persistence Tools - JPA Support
    -m2e-wtp - JPA configurator for WTP (Optional)

  5)설치 후 STS Restart

3. JPA Project 생성
  1)Package Explorer > right-click > Maven Project
    -Next
    -org.apache.maven.archetypes, maven-archetype-quickstart, 1.4
    -Next

  2)각 항목 선택 후 Finish
    -Group Id : com.example
    -artifact Id : jpademo
    -Version : 0.0.1-SNAPSHOT
    -Package : com.example.jpademo
    -Finish

  3)Project Facets 변환
    -jpademo Project > right-click > Properties > Project Facets
    -Java 1.8 > Runtimes > Apache Tomcat v9.0, jdk1.8.0_241 Check
    -Check JPA
    -만일 설정 화면 하단의 [Further configuration required...] Link에 Error message가 뜨는 경우
      --Link click
      --[JPA Facet] 창에서
        ---Platform : Generic 2.1
        ---JPA implementation
          Type : Disable Library Configuration
      --OK
    -Apply and Close

  4)Maven Project를 JPA Project로 변경하면 src/main/java하위에 META-INF/persistence.xml JPA 환경설정 파일이 생긴다.

  5)jpademo Project의 Perspective를 JPA Perspective로 변경하려면
    -Window > Perspective > Open Perspective > Other
    -JPA 선택 > Open

4. 의존성 추가
  1)pom.xml을 수정
    -Mvnrepository에서 'hibernate'로 검색하여 'Hibernate EntityManager Relocation'로 들어간다.
    -5.4.12.Final Click

      <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager -->
      <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-entitymanager</artifactId>
        <version>5.4.12.Final</version>

```
        </dependency>

    -'h2'로 검색하여 'H2 Database Engine'으로 들어가서 1.4.200 선택
        <!-- https://mvnrepository.com/artifact/com.h2database/h2 -->
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>1.4.200</version>
            <!--<scope>test</scope> --> 주의할 것, 이 scope tag는 반드시 삭제할 것
        </dependency>

    -'lombok'으로 검색하여
        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.12</version>
            <scope>provided</scope>
        </dependency>

    -pom.xml > right-click > Maven install
        [INFO] BUILD SUCCESS


5. Entity Class 작성 및 Table Mapping
  1)Table을 준비한다.
  2)JPA는 Table이 없으면 Java Class를 기준으로 Mapping할 Table을 자동으로 생성한다.
  3)Table과 Mapping되는 Java Class를 Entity라고 한다.
  4)JPA를 사용하는 데 있어서 가장 먼저 해야 할 일은 Entity를 생성하는 것이다.
  5)Value Object Class처럼 Table과 동일한 이름을 사용하고 Column과 Mapping 될 Member Variable을 선언하면 된다.
  6)다만, Eclipse의 JPA Perspective가 제공하는 Entity 생성 기능을 사용하면 Entity를 생성함과 동시에 영속성 설정 파일(persistence.xml)에 자동으로 Entity가 등록된다.
  7)src/main/java Folder에 com.example.jpademo package > right-click > New > JPA Entity
  8)다음 항목의 값을 입력 후 Finish 클릭
    -Java package : com.example.domain
    -Class name : User

  9)META-INF/persistence.xml 파일이 자동으로 수정되었다.
    <?xml version="1.0" encoding="UTF-8"?>
    <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
      http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
      <persistence-unit name="jpademo">
        <class>com.example.domain.User</class>
      </persistence-unit>
    </persistence>

  10)이제 방금 생성한 User Class를 수정한다.

    package com.example.domain;

    import java.io.Serializable;

    import javax.persistence.Entity;
    import javax.persistence.Id;
    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
1463    11)다음은 JPA 사용하는 주요 Annotation을 설명한 것이다.
1464      -@Entity
1465        --Entity Class 임을 설명
1466        --기본적으로 Class의 이름과 동일한 Table과 Mapping된다.
1467      -@Table
1468        --Entity의 이름과 Table의 이름이 다를 경우, name 속성을 이용하여 Mapping 한다.
1469        --이름이 동일하면 생략 가능
1470      -@Id
1471        --Table의 primary key와 Mapping한다.
1472        --Entiry의 필수 Annotation으로서 @Id가 없으면 Entity는 사용 불가
1473      -@GeneratedValue
1474        --@Id가 선언된 Field에 기본 키 값을 자동으로 할당
1475
1476
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"/>
```

```
            <property name="javax.persistence.jdbc.password" value="javah2"/>
            <property name="javax.persistence.jdbc.url" value="jdbc:h2:~/test"/>
            <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
            <!-- Option 속성 -->
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.format_sql" value="true"/>
            <property name="hibernate.use_sql_comments" value="false"/>
            <property name="hibernate.id.new_generator_mappings" value="true"/>
            <property name="hibernate.hbm2ddl.auto" value="create"/>
          </properties>
        </persistence-unit>
      </persistence>
```

2)여기서 중요한 속성은 hibernate.dialect 이다.
3)이 속성은 JPA 구현체가 사용할 Dialect Class를 지정할 때 사용한다.
4)이 속성을 H2Dialect Class로 설정하면 H2용 SQL이 생성되고, OracleDialect로 변경하면 Oracle용 SQL이 생성된다.

7. JPA로 Data 처리하기
1)User 등록
-src/main/java Folder에 JPAClient Class를 생성한다.

```
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

import com.example.domain.User;

public class JPAClient {
  public static void main(String[] args) {
    // EntityManager 생성
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
    EntityManager em = emf.createEntityManager();
    try {
      User user = new User();
      user.setUserid("jimin");
      user.setUsername("한지민");
      user.setGender("여");
      user.setAge(24);
      user.setCity("서울");
      // 등록
      em.persist(user);
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      em.close();
      emf.close();
    }
  }
}
```

-JPA는 META-INF/persistence.xml을 먼저 loading한다.
-그리고 persistence.xml의 unit name으로 설정한 영속성 unit 정보를 이용하여 EntityManagerFactory 객체를 생성한다.
-JPA를 이용하여 CRUD를 하려면 EntityManager 객체를 사용해야 한다.
-이것은 EntityManagerFactory를 통해 생성되며, EntityManager를 얻었으면 persist() 를 통해 User Table에 저장한다.

2)실행하기
-JPAClient > right-click > Run As > Java Application
-만일 아래의 Error Message가 나오면
ERROR: Database may be already in use: null. Possible solutions: close all other connection(s); use the server mode [90020-200]
-작업관리자에서 javaw.exe 작업끝내기를 수행한다.

3)실행 결과

```
Hibernate:

drop table User if exists

Hibernate:

  drop sequence if exists hibernate_sequence
Hibernate: create sequence hibernate_sequence start with 1 increment by 1

Hibernate:
  create table User (
    userid varchar(255) not null,
    age integer not null,
    city varchar(255),
    gender varchar(255),
    username varchar(255),
    primary key (userid)
  )
```

4)하지만 실제 Database에는 Data가 Insert되지 않았다.

8. Transaction 관리

1)JPA가 실제 Table에 등록/수정/삭제 작업을 처리하기 위해서는 해당 작업이 반드시 Transaction안에서 수행되어야 한다.
2)만약 Transaction을 시작하지 않았거나 등록/수정/삭제 작업 이후에 Transaction을 종료하지 않으면 요청한 작업이 실제 Database에 반영되지 않는다.
3)JPAClient.java를 수정한다.

```
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

import com.example.domain.User;

public class JPAClient {
  public static void main(String[] args) {
    // EntityManager 생성
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
    EntityManager em = emf.createEntityManager();
    //Transaction 생성
    EntityTransaction tx = em.getTransaction();
    try {
      //Transaction 시작
      tx.begin();

      User user = new User();
```

```
            user.setUserid("jimin");
            user.setUsername("한지민");
            user.setGender("여");
            user.setAge(24);
            user.setCity("서울");
            // 등록
            em.persist(user);

            // Transaction commit
            tx.commit();
          } catch (Exception e) {
            e.printStackTrace();
            // Transaction rollback
            tx.rollback();
          } finally {
            em.close();
            emf.close();
          }
        }
      }
```

4)실행하기
-JPAClient > right-click > Run As > Java Application

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

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

9. 데이터 누적하기
1)현재 작성한 JPA 프로그램은 아무리 많이 실행해도 한 건의 Data만 등록된다.
2)즉 매번 Table이 새롭게 생성되기 때문이다.
3)따라서 JPA Client를 실행할 때마다 Data를 누적하기 위해서는 persistence.xml에서 다음을 수정해야 한다.

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

4)다음으로 변경한다.

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

5)이렇게 변경하면 새롭게 생성하지 않고 기본의 Table을 재사용한다.
6)다음의 Data를 수행해서 3명의 User를 Insert한다.

chulsu, 34, 부산, 남, 김철수
younghee, 44, 대전, 여, 이영희

10. Data 검색
1)JPAClient.java를 수정한다

```
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
```

```
import javax.persistence.Persistence;

import com.example.domain.User;

public class JPAClient {
  public static void main(String[] args) {
    // EntityManager 생성
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
    EntityManager em = emf.createEntityManager();
    try {
      // User 검색
      User user = em.find(User.class, "jimin");
      System.out.println("jimin --> " + user);
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      em.close();
      emf.close();
    }
  }
}
```

2)실행

```
Hibernate:
  select
    user0_.userid as userid1_0_0_,
    user0_.age as age2_0_0_,
    user0_.city as city3_0_0_,
    user0_.gender as gender4_0_0_,
    user0_.username as username5_0_0_
  from
    User user0_
  where
    user0_.userid=?
jimin --> User [userid=jimin, username=한지민, gender=여, age=24, city=서울]
```

11. Entity 수정

1)JPAClient.java 수정

```
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

import com.example.domain.User;

public class JPAClient {
  public static void main(String[] args) {
    // EntityManager 생성
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
    EntityManager em = emf.createEntityManager();
    // Transaction 생성
    EntityTransaction tx = em.getTransaction();
    try {
      // Transaction 시작
      tx.begin();
```

```
            // 수정할 User 조회
            User user = em.find(User.class, "younghee");
            user.setCity("광주");
            user.setAge(55);
            // Transaction commit
            tx.commit();
          } catch (Exception e) {
            e.printStackTrace();
            // Transaction rollback
            tx.rollback();
          } finally {
            em.close();
            emf.close();
          }
        }
      }
```

2)실행

```
    Hibernate:
      select
        user0_.userid as userid1_0_0_,
        user0_.age as age2_0_0_,
        user0_.city as city3_0_0_,
        user0_.gender as gender4_0_0_,
        user0_.username as username5_0_0_
      from
        User user0_
      where
        user0_.userid=?
    Hibernate:
      update
        User
      set
        age=?,
        city=?,
        gender=?,
        username=?
      where
        userid=?
```

12. Entity 삭제

1)JPAClient.java 수정

```
    import javax.persistence.EntityManager;
    import javax.persistence.EntityManagerFactory;
    import javax.persistence.EntityTransaction;
    import javax.persistence.Persistence;

    import com.example.domain.User;

    public class JPAClient {
      public static void main(String[] args) {
        // EntityManager 생성
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
        EntityManager em = emf.createEntityManager();
        // Transaction 생성
        EntityTransaction tx = em.getTransaction();
```

```
        try {
          // Transaction 시작
          tx.begin();

          // 삭제할 User 조회
          User user = em.find(User.class, "younghee");
          em.remove(user);

          // Transaction commit
          tx.commit();
        } catch (Exception e) {
          e.printStackTrace();
          // Transaction rollback
          tx.rollback();
        } finally {
          em.close();
          emf.close();
        }
      }
    }
```

2)실행

```
    Hibernate:
      select
        user0_.userid as userid1_0_0_,
        user0_.age as age2_0_0_,
        user0_.city as city3_0_0_,
        user0_.gender as gender4_0_0_,
        user0_.username as username5_0_0_
      from
        User user0_
      where
        user0_.userid=?
    Hibernate:
      delete
      from
        User
      where
```

13. 여러 Record 조회와 JPQL

1)한 건의 Record 조회는 find()를 사용한다.

2)하지만, 여러 건의 Record를 조회하기 위해서는 JPQL(Java Persistence Query Language)라는 JPA에서 제공하는 별도의 Query 명령어를 사용해야 한다.

3)JPAClient.java 수정

```
    import java.util.List;

    import javax.persistence.EntityManager;
    import javax.persistence.EntityManagerFactory;
    import javax.persistence.EntityTransaction;
    import javax.persistence.Persistence;

    import com.example.domain.User;

    public class JPAClient {
      public static void main(String[] args) {
        // EntityManager 생성
```

```
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
        EntityManager em = emf.createEntityManager();

        // Transaction 생성
        EntityTransaction tx = em.getTransaction();

        try {
          // Transaction 시작
          tx.begin();

          User user = new User();
          user.setUserid("hojune");
          user.setUsername("이호준");
          user.setAge(30);
          user.setGender("남");
          user.setCity("수원");

          // User 등록
          em.persist(user);

          // Transaction commit
          tx.commit();

          // 여러 Record 조회
          String jpql = "SELECT u FROM User u ORDER BY u.userid DESC";
          List<User> userList = em.createQuery(jpql, User.class).getResultList();
          for (User usr : userList) {
            System.out.println(usr);
          }
        } catch (Exception e) {
          e.printStackTrace();
          // Transaction rollback
          tx.rollback();
        } finally {
          em.close();
          emf.close();
        }
      }
    }

4)실행
  Hibernate:
    insert
    into
      User
      (age, city, gender, username, userid)
    values
      (?, ?, ?, ?, ?)
  Hibernate:
    select
      user0_.userid as userid1_0_,
      user0_.age as age2_0_,
      user0_.city as city3_0_,
      user0_.gender as gender4_0_,
      user0_.username as username5_0_
    from
      User user0_
    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
```

3)src/main/resources/static/css folder 생성
-bootstrap.min.css 추가할 것

4)src/main/resources/static/index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Home page</title>
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css" />
<script src="js/jquery-3.5.0.min.js"></script>
<script>
  $(document).ready(function() {
    alert("Hello, Spring Boot World!!!");
  });
</script>
</head>
<body>
  <div>
    <img src="images/spring-boot.png" />
  </div>
  <div class="jumbotron">
    <h1>Hello, Spring Boot World</h1>
    <p>...</p>
    <p>
      <a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>
    </p>
  </div>
</body>
</html>
```

4. Spring Boot에서는 template을 사용하지 않을 경우 기본적으로 static file은 src/main/resources/static에서 찾는다.
5. 이럴 때는 반드시 file의 확장자 .html까지 넣어야 한다.
6. springweb Project > right-click > Run As > Spring Boot App
7. http://localhost:8080/

Task9. JSP Page 만들기
1. Spring Boot에서는 JSP 사용을 권장하지 않는다.
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.
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.
4. 그래서 template을 통해 code를 분리해야 할 필요가 있는 것이다.

5. Spring Boot project 생성
1)Package Explorer > right-click > New > Spring Starter Project

2)다음 각 항목의 값을 입력 후 Next 클릭
-Service URL :http://start.spring.io
-Name : springjspdemo
-Type : Maven
-Packaging : jar

-Java Version : 8
-Language : Java
-Group : com.example
-Artifact : springjspdemo
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.biz

3)각 항목 선택 후 Finish 클릭
-Spring Boot Version : 2.2.6
-Web > Spring Web > Finish

6. pom.xml
1)jstl을 위해

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
```

2)JSP를 위해

```
<dependency>
  <groupId>org.apache.tomcat</groupId>
  <artifactId>tomcat-jasper</artifactId>
  <version>9.0.33</version>
</dependency>
```

3)pom.xml > right-click > Run As > Maven install
[INFO] BUILD SUCCESS

7. folder 준비
1)src/main folder 안에 webapp folder 생성
2)webapp folder 안에 WEB-INF folder 생성
3)WEB-INF folder 안에 jsp folder 생성

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

9. src/main/resources/application.properties code 추가
1)application.properties > right-click > Open with > Generic Editor
spring.mvc.view.prefix : /WEB-INF/jsp/
spring.mvc.view.suffix : .jsp

10. jsp folder 안에 jsp file 생성
1)index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="java.util.Date, java.text.SimpleDateFormat" %>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
  </head>
  <body>
```

```
        <h1>Index page</h1>
        <%=new SimpleDateFormat("yyyy년 MM월 dd일").format(new Date()) %>
      </body>
    </html>
```

11. HomeController class 생성
  1)com.example.biz > right-click > New > Click
  2)Name : HomeController
  3)Finish

```
    package com.example.biz;

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

    @Controller
    public class HomeController {

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

12. 실행
  http://localhost:8080/

```
    Index page
    2020년 04월 20일
```

Task10. thymeleaf template 사용하기
1. Spring Boot project 생성
  1)Package Explorer > right-click > New > Spring Starter Project
  2)다음의 각 항목 입력 후 Next 클릭
    -Service URL :http://start.spring.io
    -Name : MyBootWeb
    -Type : Maven
    -Packaging : jar
    -Java Version : 8
    -Language : Java
    -Group : com.example
    -Artifact : MyBootWeb
    -Version : 0.0.1-SNAPSHOT
    -Description : Demo project for Spring Boot
    -Package : com.example.biz

  3)각 항목 선택 후 Finish 클릭
    -Spring Boot Version : 2.2.6
    -Web > Spring Web > Finish

2. src/main/java/com.example.biz.MyBootWebApplication.java

```
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) {
```

```
        int result = 0;
        for(int i = 1 ; i <= num ; i++) result += i;
        return "total : " + result;
      }


8. project > right-click > Run As > Spring Boot App

9. http://localhost:8080/100

   total : 5050


10. pom.xml에 lombok dependency 추가

   <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.12</version>
    <scope>provided</scope>
   </dependency>

   -pom.xml > right-click > Run As > Maven install
     [INFO] BUILD SUCCESS


11. 객체를 JSON으로 출력하기
   1)src/main/java/com.example.biz/Student.java 생성

     package com.example.biz;

     import lombok.AllArgsConstructor;
     import lombok.Data;

     @Data
     @AllArgsConstructor
     public class Student {
        private int userid;
        private String name;
        private int age;
        private String address;
     }

   2)HelloController.java 수정

     @RestController
     public class HelloController {
        String [] names = {"조용필", "이미자", "설운도"};
        int [] ages = {56, 60, 70};
        String [] addresses = {"서울특별시", "부산광역시", "대전광역시"};

        @RequestMapping("/{userid}")
        public Student index(@PathVariable int userid) {
           return new Student(userid, names[userid], ages[userid], addresses[userid]);
        }
     }

   3)http://localhost:8080/1
```

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


11. thymeleaf 추가하기
  1)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>

  2)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";
      }
    }

  3)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; }
          body { font-size:13pt; color:gray; margin:5px 25px; }
        </style>
      </head>
      <body>
        <h1>Hello! Spring Boot with Thymeleaf</h1>
        <p>This is sample web page.</p>
      </body>
    </html>

  4)http://localhost:8080/

    Hello! Spring Boot with Thymeleaf
    This is sample web page
```

5)template에 값 표시하기
-index.html code 수정

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

-HelloController.java 수정

```
@Controller
public class HelloController {

  @RequestMapping("/{num}")
  public String index(@PathVariable int num, Model model) {
    int result = 0;
    for(int i = 1 ; i <= num ; i++) result += i;
    model.addAttribute("msg", "total : " + result);
    return "index";
  }
}
```

6)http://localhost:8080/100

Hello! Spring Boot with Thymeleaf
total : 5050

7)ModelAndView class 사용하기
-HelloController.java 수정

```
@Controller
public class HelloController {
  @RequestMapping("/{num}")
  public ModelAndView index(@PathVariable int num, ModelAndView mav) {
    int result = 0;
    for(int i = 1 ; i <= num ; i++) result += i;
    mav.addObject("msg", "total : " + result);
    mav.setViewName("index");
    return mav;
  }
}
```

8)http://localhost:8080/100

Hello! Spring Boot with Thymeleaf
total : 5050

9)form 사용하기
-index.html 수정

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Index Page</title>
    <style type="text/css">
      h1 { font-size:18pt; font-weight:bold; color:gray; }
      body { font-size:13pt; color:gray; margin:5px 25px; }
    </style>
```

```
        </head>
        <body>
          <h1>Hello!</h1>
          <p th:text="${msg}"></p>
          <form method="post" action="/send">
            <input type="text" name="txtName" th:value="${value}" />
            <input type="submit" value="Send" />
          </form>
        </body>
      </html>
```

-HelloController.java 수정

```
      @Controller
      public class HelloController {

        @RequestMapping(value = "/", method = RequestMethod.GET)
        public ModelAndView index(ModelAndView mav) {
          mav.setViewName("index");
          mav.addObject("msg", "Please write your name...");
          return mav;
        }

        @RequestMapping(value = "/send", method = RequestMethod.POST)
        public ModelAndView send(@RequestParam("txtName") String name, ModelAndView
        mav) {
          mav.addObject("msg", "안녕하세요! " + name + "님!");
          mav.addObject("value", name);
          mav.setViewName("index");
          return mav;
        }
      }
```

10)http://localhost:8080

11)기타 form controller

-index.html 수정

```
    <!doctype html>
    <html lang="en">
      <head>
        <meta charset="UTF-8" />
        <title>Index Page</title>
        <style type="text/css">
          h1 {
            font-size: 18pt;
            font-weight: bold;
            color: gray;
          }
          body {
            font-size: 13pt;
            color: gray;
            margin: 5px 25px;
          }
        </style>
      </head>
      <body>
      <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
            </label>
2411        </div>
2412        <div>
2413          <input type="radio" id="male" name="gender" value="male" /> <label for="male">
            남성</label>
2414        </div>
2415        <div>
2416          <input type="radio" id="female" name="gender" value="female" /> <label
            for="female">여성</label>
2417        </div>
2418        <div>
2419          <select name="selOs" size="4">
2420            <option>--선택--</option>
2421            <option value="Windows">windows</option>
2422            <option value="MacOS">MacOS</option>
2423            <option value="Linux">Linux</option>
2424          </select>
2425          <select name="selEditors" size="4" multiple="multiple">
2426            <option>--선택--</option>
2427            <option value="Notepad">Notepad</option>
2428            <option value="Editplus">Editplus</option>
2429            <option value="Visual Studio Code">Visual Studio Code</option>
2430            <option value="Sublime Text">Sublime Text</option>
2431            <option value="Eclipse">Eclipse</option>
2432          </select>
2433        </div>
2434        <input type="submit" value="전송" />
2435        </form>
2436      </body>
2437    </html>
2438
2439    12)HelloController.java 수정
2440
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
          check1,
2454              @RequestParam(value="gender", required=false) String gender,
2455              @RequestParam(value="selOs", required=false) String selOs,
2456              @RequestParam(value="selEditors", required=false) String [] selEditors,
                ModelAndView mav) {
2457          String result = "";
2458          try {
2459            result = "check : " + check1 + ", gender : " + gender + ", OS : " + selOs +
```

```
            "\nEditors : ";
        }catch(NullPointerException ex) {}

        try {
          result += selEditors[0];
          for(int i = 1 ; i < selEditors.length ; i++) result += ", " + selEditors[i];
        }catch(NullPointerException ex) {
          result += "null";
        }

        mav.addObject("msg", result);
        mav.setViewName("index");
        return mav;
      }
    }
```

13)http://localhost:8080

12. Redirect

1)index.html 수정

```
    <body>
      <h1>Hello! index.</h1>
    </body>
```

2)HelloController.java 수정

```
    @Controller
    public class HelloController {

      @RequestMapping("/")
      public ModelAndView index(ModelAndView mav) {
        mav.setViewName("index");
        return mav;
      }

      @RequestMapping("/other")
      public String other() {
        return "redirect:/";
      }

      @RequestMapping("/home")
      public String home() {
        return "forward:/";
      }
    }
```

3)redirect와 forward 차이점 구분하기

Task11. thymeleaf template 사용하기2

1. Spring Boot project 생성

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

2)다음 각 항목의 값을 입력 후 Next 클릭

-Service URL :http://start.spring.io
-Name : templatedemo
-Type : Maven
-Packaging : Jar
-Java Version : 8
-Language : Java
-Group : com.example
-Artifact : templatedemo
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.biz

3)다음 각 항목 선택 후 Finish 클릭
-Spring Boot Version : 2.2.6
-Select
--Developer Tools > Spring Boot DevTools, Lombok
--Web > Spring Web
--Template Engines > Thymeleaf
-Finish

2. HomeController.java 생성
1)com.example.biz > right-click > New > Class

2)Name : HomeController

```
package com.example.biz;

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

@Controller
public class HomeController {

  @GetMapping("/")
  public String home() {
    return "index";
  }
}
```

3)index.html 생성
-src/main/resources/templates > New > Other > Web > HTML File > Next
-File name : index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Insert title here</title>
  </head>
  <body>
    <h1>Hello Page</h1>
    <p th:text="${new java.util.Date().toString()}"></p>
  </body>
</html>
```

4)실행
--http://localhost:8080

Hello Page
Fri Feb 21 00:31:37 KST 2020

3. Utility Object 사용하기
1)index.html 수정

```
<body>
  <h1>Hello Page</h1>
  <p th:text="${#dates.format(new java.util.Date(), 'dd/MM/yyyy HH:mm')}"></p>
  <p th:text="${#numbers.formatInteger(1234,7)}"></p>
  <p th:text="${#strings.toUpperCase('Welcome to Spring.')}"></p>
</body>
```

2)실행

Hello Page

09/11/2019 00:15

0001234

WELCOME TO SPRING

4. 매개변수에 접근하기
1)index.html의 <body> 부분을 아래와 같이 수정한다.

```
<body>
  <h1>Hello page</h1>
  <p th:text="'from parameter... id=' + ${param.id[0] + ',name=' +
  param.name[0]}"></p>
</body>
```

2)HomeController.java를 수정한다.

```
@RequestMapping("/")
public ModelAndView index(ModelAndView mav) {
  mav.setViewName("index");
  return mav;
}

@RequestMapping("/home")
public String home() {
  return "forward:/";
}
```

3)그리고 id와 name을 query string으로 지정해서 접속한다.
-http://localhost:8080/home/?id=javaexpert&name=Springboot
-결과는 아래와 같다.
Helo page
from parameter... id=javaexpert,name=Springboot

4)controller를 거치지 않고 template내에서 직접 전달된 값을 사용할 수 있다.
5)중요한 것은 param내의 id나 name 배열에서 첫 번째 요소를 지정해서 추출한다는 것이다.
6)그 이유는 query string으로 값을 전송할 때 같은 이름의 값을 여러 개 전송하기 위해서다.
7)예를 들면, http://localhost:8080/home/?id=123&id=456&name=javaexpert&name=peter
8)이렇게 하면 param에서 추출하는 id와 name이 각각 {123,456}, {javaexpert,peter}가 되는 것이기 때문

이다.
9)여기서 사용하고 있는 th:text의 값을 보면 큰따옴표 안에 다시 작은 따옴표를 사용해서 값을 작성하고 있다.
10)이것은 OGNL로 text literal을 작성할 때 사용하는 방식이다.
11)이렇게 하면 다수의 literal을 연결할 때 큰 따옴표안에 작은 따옴표 literal을 사용할 수 있게 된다.

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

5. Message식 사용하기
1)src/main/resources > right-click > New > File
2)File name : messages.properties > Finish

```
content.title=Message sample page.
content.message=This is sample message from properties.
```

3)index.html 수정

```
<body>
<h1 th:text="#{content.title}">Hello page</h1>
<p th:text="#{content.message}"></p>
</body>
```

4)실행

```
Message sample page.
This is sample message from properties.
```

6. Link식과 href
1)index.html

```
<body>
  <h1 th:text="#{content.title}">Helo page</h1>
  <p><a th:href="@{/home/{orderId}(orderId=${param.id[0]})}">link</a></p>
</body>
```

2)접속할 때 query string에 id를 지정한다.
3)예를 들어 http://localhost:8080/?id=123에 접속하면 link에는 /home/123이 설정된다.

7. 선택 객체와 변수식
1)src/main/java/com.example.biz > right-click > New > Class
2)Name : Member

```
package com.example.biz;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class Member {
  private int id;
  private String username;
  private int age;
}
```

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
```

3)다음 각 항목을 선택후 Finish 클릭
-Spring Boot Version : 2.2.6
-Select
--SQL > check MySQL, JDBC API
--Web > Spring Web
--Developer Tools > Spring Boot DevTools, Lombok
-Finish

4)위에서 type을 선택시 고려사항
-만일 Embeded된 tomcat으로 Stand-Alone형태로 구동시키기 위한 목적이라면 Packaging Type을 jar로 선택한다.
-war로 선택할 경우, 기존과 같이 외부의 tomcat으로 deploy 하는 구조로 만들어진다.
-물론, source의 최종배포의 형태가 server의 tomcat에 deploy해야 하는 구조라면 처음부터 war로 만들어서 작업해도 상관없다.
-jar로 선택하고, local에서 개발 및 test를 하다가 나중에 배포할 경우 war로 변경해서 배포를 할 수도 있다.

2. pom.xml
1)jstl 사용을 위한

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
```

2)jasper 사용을 위한

```
<dependency>
  <groupId>org.apache.tomcat</groupId>
  <artifactId>tomcat-jasper</artifactId>
  <version>9.0.33</version>
</dependency>
```

3)MariaDB 사용을 위한

```
<dependency>
  <groupId>org.mariadb.jdbc</groupId>
  <artifactId>mariadb-java-client</artifactId>
  <version>2.5.4</version>
</dependency>
```

4)pom.xml > right-click > Run As > Maven install
[INFO] BUILD SUCCESS

3. src/main/resources/application.properties

```
spring.application.name=BootJDBCDemo
spring.datasource.url=jdbc:mariadb://localhost:3306/test
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=javamariadb
```

4. Table 생성

```
CREATE TABLE test.User
(
  username VARCHAR(20) PRIMARY KEY,
  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 > Inteface
2831    4)Name : UserDao
2832
2833      package com.example.dao;
2834
2835      import java.util.List;
2836
2837      import com.example.vo.UserVO;
2838
2839      public interface UserDao {
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      }
2948
2949
2950  8. static resources 준비
2951    1)src/main/resources/static/images folder 생성
2952      -spring-boot.png 추가할 것
2953
2954    2)src/main/resources/static/js folder 생성
2955      -jquery-3.4.1.min.js 추가할 것
2956
2957    3)src/main/resources/static/css folder 생성
2958      -style.css
2959
2960      @charset "UTF-8";
2961      body {
2962        background-color:yellow;
2963      }
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
```

7)만일 JSP를 template으로 사용하는 경우에는 이 WEB-INF folder 안에 template file을 준비할 필요가 있다.
8)folder를 다 만들었으면, WEB 루트(Context Root)로 인식할 있도록 환경설정을 아래와 같이 추가한다.
-src/main/resources/application.properties code 추가

```
spring.mvc.view.prefix : /WEB-INF/jsp/
spring.mvc.view.suffix : .jsp
```

10. jsp folder 안에 jsp file 생성
1)index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>New User Insertion Page</title>
    <link rel="stylesheet" type="text/css" href="/css/style.css">
    <c:url var="jsurl" value="/js/jquery-3.3.1.slim.min.js" />
    <script src="${jsurl}"></script>
    <script>
      $(document).ready(function() {
        alert("Hello, Spring Boot!");
      });
    </script>
  </head>
  <body>
    <img src="/images/spring-boot.png" />
    <h1>New User Insertion Page</h1>
    <form action="/user" method="post">
      Name : <input type="text" name="username" /><br />
      Age : <input type="number" name="age" /><br />
      <button type="submit">Submit</button>
    </form>
  </body>
</html>
```

2)list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>User List</title>
    <link rel="stylesheet" type="text/css" href="/css/style.css">
  </head>
  <body>
    <h1>User List</h1>
    <ul>
      <c:forEach var="user" items="${users}">
        <li>${user.username}(${user.age})</li>
      </c:forEach>
    </ul>
  </body>
</html>
```

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

```
package com.example.biz;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan("com.example")    <-- 추가 code
public class BootJdbcDemoApplication {

  public static void main(String[] args) {
    SpringApplication.run(BootJdbcDemoApplication.class, args);
  }
}
```

-@SpringBootConfiguration 은 다음의 annotation 3개를 모두 담고 있다.
  --@SpringBootConfiguration
  --@EnableAutoConfiguration
  --@ComponentScan
-만약, AutoScan이 되어야 하는 component class들 - 대표적으로 @Controller, @Service, @Repository, @Component등-의 위치가 main class가 위치한 package보다 상위 package에 있거나, 하위가 아닌 다른 package에 있는 경우, scan이 되지 않는다.
-Controller class가 main의 하위 class에 있으면 상관없지만, 예를 들어, main class가 다른 package나 하위 package가 아니면 아래와 같이 해줘야 한다.
-명시적으로 ComponentScan을 할 Base Package를 지정해주면 된다.
  --ex) @ComponentScan(basePackages = "com.springboot.demo")

14)project > right-click > Run As > Spring Boot App
15)http://localhost:8080

Task13. String Jdbc API 사용하기
1. Spring Boot project 생성
  1)Package Explorer > right-click > New > Spring Starter Project
  2)다음의 각 항목의 값을 입력후 Next 클릭
    -Service URL :http://start.spring.io
    -Name : SpringBootJdbcDemo
    -Type : Maven
    -Packaging : Jar
    -Java Version : 8
    -Language : Java
    -Group : com.example
    -Artifact : SpringBootJdbcDemo
    -Version : 0.0.1-SNAPSHOT
    -Description : Demo project for Spring Boot
    -Package : com.example.biz

  3)다음 각 항목을 선택후 Finish 클릭
    -Spring Boot Version : 2.2.6
    -Select
      --SQL > check Oracle Driver, JDBC API
      --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 > Inteface
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(),
            member.getAge());
3186        }
3187
3188        private class MyRowMapper implements RowMapper<MemberVO> {
3189          @Override
3190          public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
3191            return new MemberVO(rs.getString("userid"), rs.getString("username"),
              rs.getInt("age"));
3192          }
3193        }
3194
3195        @Override
3196        public List<MemberVO> readAll() {
3197          return this.jdbcTemplate.query("SELECT * FROM Member ORDER BY userid DESC",
            new MyRowMapper());
3198        }
3199
```

```
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        }
```

```
    }


7. static resources 준비
  1)src/main/resources/static/images folder 생성
    -spring-boot.png 추가할 것
  2)src/main/resources/static/js folder 생성
    -jquery-3.5.0.min.js 추가할 것
  3)src/main/resources/static/css folder 생성
    -style.css

      @charset "UTF-8";
      body {
        background-color:yellow;
      }
      h1{
        color : blue;
      }


8. templates/index.html 생성

  <!DOCTYPE html>
  <html>
  <head>
  <meta charset="UTF-8" />
  <title>New User Insertion Page</title>
  <link rel="stylesheet" type="text/css" href="/css/style.css">
  <script src="/js/jquery-3.5.0.min.js"></script>
  <script>
    $(document).ready(function() {
      alert("Hello, Spring Boot!");
    });
  </script>
  </head>
  <body>
    <img src="/images/spring-boot.jpg" />
    <h1>New User Insertion Page</h1>
    <form action="/member" method="post">
      <ul>
        <li>ID : <input type="text" name="userid" /></li>
        <li>Name : <input type="text" name="username" /></li>
        <li>Age : <input type="number" name="age" /></li>
        <li><button>Submit</button></li>
      </ul>
    </form>
  </body>
  </html>


9. templates/list.html 생성

  <!DOCTYPE html>
  <html xmlns:th="http://www.thymeleaf.org">
  <head>
  <meta charset="UTF-8" />
  <title>회원정보</title>
  <style>
```

```
    h1 {
      font-size: 18pt;  font-weight: bold; color: gray;
    }

    body {
      font-size: 13pt;    color: gray;    margin: 5px 25px;
    }

    tr {
      margin: 5px;
    }

    th {
      padding: 5px; color: white; background: darkgray;
    }

    td {
      padding: 5px; color: black; background: #e0e0ff;
    }
  </style>
  </head>
  <body>
    <h1>회원명부</h1>
    <table border='1'>
      <thead>
        <tr>
          <th>아이디</th>
          <th>이름</th>
          <th>나이</th>
        </tr>
      </thead>
      <tbody>
        <tr th:each="member : ${members}">
          <td th:text="${member.userid}"></td>
          <td th:text="${member.username}"></td>
          <td th:text="${member.age}"></td>
        </tr>
      </tbody>
    </table>
  </body>
  </html>


10. src/main/java/com.example.biz/SpringBootJdbcDemoApplication.java

  package com.example.biz;

  import org.springframework.boot.SpringApplication;
  import org.springframework.boot.autoconfigure.SpringBootApplication;
  import org.springframework.context.annotation.ComponentScan;

  @SpringBootApplication
  @ComponentScan("com.example")
  public class SpringBootJdbcDemoApplication {

    public static void main(String[] args) {
      SpringApplication.run(SpringBootJdbcDemoApplication.class, args);
    }
```

```
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,
```

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

  TRUNCATE TABLE Member;


3. src/main/resources/application.properties
  spring.datasource.url=jdbc:oracle:thin:@localhost:1521:XE    <--반드시 넣을 것, 매우 주의
  spring.datasource.hikari.driver-class-name=oracle.jdbc.driver.OracleDriver
  spring.datasource.hikari.jdbc-url=jdbc:oracle:thin:@localhost:1521:XE
  spring.datasource.hikari.username=membership
  spring.datasource.hikari.password=membership
  spring.datasource.hikari.connection-timeout=20000
  spring.datasource.hikari.minimum-idle=5
  spring.datasource.hikari.maximum-pool-size=20
  spring.datasource.hikari.idle-timeout=300000
  spring.datasource.hikari.max-lifetime=1200000
  spring.datasource.hikari.auto-commit=true


4. DataSource 설정하기
  1)src/main/java > right-click > New > package
  2)Name : com.example.config
  3)Finish
  4)com.example.config > right-click > New > Class
  5)Name : DatabaseConfiguration
  6)Finish

  package com.example.config;

  import javax.sql.DataSource;

  import org.springframework.boot.context.properties.ConfigurationProperties;
  import org.springframework.context.annotation.Bean;
  import org.springframework.context.annotation.Configuration;
  import org.springframework.context.annotation.PropertySource;

  import com.zaxxer.hikari.HikariConfig;
  import com.zaxxer.hikari.HikariDataSource;

  import lombok.extern.slf4j.Slf4j;

  @Slf4j
  @Configuration
  @PropertySource("classpath:application.properties")
  public class DatabaseConfiguration {

    @Bean
    @ConfigurationProperties(prefix="spring.datasource.hikari")
    public HikariConfig hikariConfig() {
      return new HikariConfig();
    }
```

```
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]
         com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
3524      2020-04-21 23:50:38.037  WARN 8420 --- [  restartedMain]
         com.zaxxer.hikari.util.DriverDataSource  : Registered driver with
         driverClassName=oracle.jdbc.driver.OracleDriver was not found, trying direct
         instantiation.
3525      2020-04-21 23:50:38.252  INFO 8420 --- [  restartedMain]
         com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
3526      2020-04-21 23:50:38.252  INFO 8420 --- [  restartedMain]
         c.example.config.DatabaseConfiguration   : datasource : HikariDataSource (HikariPool-1)
3527
3528
3529  7. Mabatis 연동하기
3530    1)Mapper Folder 생성하기
3531      -Mapper Folder에는 Application에서 사용할 Query를 담고 있는 XML 파일을 저장한다.
3532      -src/main/resources/mapper folder 생성한다.
3533      -src/main/resources > right-click > New > Folder
3534      -Folder name : mapper
3535      -Finish
3536
3537    2)src/main/resources/mapper/mybatis-config.xml 생성
3538
3539      <?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <typeAliases>
    <typeAlias type="com.example.vo.MemberVO" alias="memberVO"/>
  </typeAliases>
</configuration>
```

3)src/main/resources/mapper/mybatis-mapper.xml 생성

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.example.vo.MemberVO">
</mapper>
```

4)com.example.config.DatabaseConfiguration.java 수정하기

```
package com.example.config;

import javax.sql.DataSource;

import org.apache.ibatis.session.SqlSessionFactory;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.io.Resource;
import org.springframework.core.io.support.PathMatchingResourcePatternResolver;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Configuration
@PropertySource("classpath:application.properties")
public class DatabaseConfiguration {
  @Autowired
  private ApplicationContext applicationContext;

  @Bean
  @ConfigurationProperties(prefix="spring.datasource.hikari")
  public HikariConfig hikariConfig() {
    return new HikariConfig();
  }

  @Bean
  public DataSource dataSource() throws Exception{
    DataSource dataSource = new HikariDataSource(hikariConfig());
    log.info("datasource : {}", dataSource);
```

```
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 sqlSessioinTemplate(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      }
```

10. MyBatis 연결 Test
-src/test/java/com.example.biz.SpringBootMembershipApplicationTests.java

```
package com.example.biz;

import org.junit.jupiter.api.Test;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@SpringBootTest
class SpringBootMembershipApplicationTests {
  @Autowired
  private SqlSessionTemplate sqlSession;

  @Test
  void contextLoads() {
  }

  @Test
  public void test1() throws Exception{
    log.info("SqlSession = " + this.sqlSession);
  }
}
```

-Run As > JUnit Test
Green Bar
2020-04-22 21:45:04.154 INFO 9628 --- [main] e.b.SpringBootMembershipApplicationTests : Started SpringBootMembershipApplicationTests in 5.494 seconds (JVM running for 8.844)
2020-04-22 21:45:04.702 INFO 9628 --- [main] e.b.SpringBootMembershipApplicationTests : SqlSession = org.mybatis.spring.SqlSessionTemplate@3f357c9d

11. MemberDao, MemberDaoImpl 생성하기
1)com.example.dao > right-click > New > Interface
2)Name : MemberDao
3)Finish

```
package com.example.dao;

import java.util.Map;

import com.example.vo.MemberVO;

public interface MemberDao {
  void create(MemberVO member);
  void readAll(Map map);
  void read(Map map);
  void update(MemberVO member);
  void delete(String userid);
}
```

4)com.example.dao > right-click > New > Class
5)Name : MemberDaoImpl
6)Interfaces : com.example.dao.MemberDao
7)Finish

12. MemberService, MemberServiceImpl 생성하기
1)com.example.service > right-click > New > Interface
2)Name : MemberService
3)Finish

```
package com.example.service;

import java.util.Map;

import com.example.vo.MemberVO;

public interface MemberService {
  void insertMember(MemberVO member);
  void selectAllMembers(Map map);
  void selectMember(Map map);
  void updateMember(MemberVO member);
  void deleteMember(String userid);
}
```

4)com.example.service > right-click > New > Class
5)Name : MemberServiceImpl
6)Interfaces : com.example.service.MemberService
7)Finish

13. css, images, js folder 생성하기
1)src/main/resources/static > right-click > New > Folder
2)Folder name : css
-bootstrap-theme.min.css
-bootstrap.min.css

3)src/main/resources/static > right-click > New > Folder
4)Folder name : images
-spring-boot.png

5)src/main/resources/static > right-click > New > Folder
6)Folder name : js
-bootstrap.min.js
-jquery-3.5.0.min.js

7)src/main/resources/static > right-click > New > Folder
8)Folder name : fonts
-glyphicons font 5개

14. src/main/resources/templates에 index.html 생성하기

```
<!DOCTYPE html>
<html lang="en" xmlns:th="https://www.thymeleaf.org/">
<head>
  <meta charset="UTF-8" />
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Welcome example.com</title>
    <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
    <link rel="stylesheet" th:href="@{/css/bootstrap-theme.min.css}" />
    <script th:src="@{js/jquery-3.5.0.min.js}"></script>
  </head>
  <body>
    <div class="jumbotron">
      <h1>Welcome to www.example.com</h1>
      <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>
      <p>
       <a class="btn btn-primary btn-lg" th:href="@{register}" role="button">회원가입</a>
      </p>
    </div>
  </body>
  </html>
```

15. MainController.java 생성

1)com.example.biz > right-click > New > Class
2)Name : MainController
3)Finish

```
  package com.example.biz;

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

  @Controller
  public class MainController {

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

4)SpringBootMembership project > right-click > Run As > Spring Boot App
5)http://localhost:8080

16. 회원가입하기

1)MainController.java 코드 추가

```
  @GetMapping("/register")
  public String register() {
    return "register";
  }
```

2)templates > right-click > New > Web > HTML File
3)Name : register.html

```
  <!DOCTYPE html>
```

```
<html lang="en" xmlns:th="https://www.thymeleaf.org/">
<head>
   <meta charset="UTF-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <title>회원가입</title>
   <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
   <script th:src="@{/js/jquery-3.5.0.min.js}"></script>
   <script>
      $(function () {
         var flag = 0;
         $("#userid").bind("change", function () {
            if ($("#userid").val()) {
               $("#useriddiv").attr("class", "form-group has-success has-feedback");
               $("#useridicon").attr("class", "glyphicon glyphicon-ok
               form-control-feedback");
               $("#useridsr").text("(success)");
               flag++;
            }
         });
         $("#passwd").bind("change", function () {
            if ($("#passwd").val()) {
               $("#passwddiv").attr("class", "form-group has-success has-feedback");
               $("#passwdicon").attr("class", "glyphicon glyphicon-ok
               form-control-feedback");
               $("#passwdsr").text("(success)");
               flag++;
            }
         });
         $("#name").bind("change", function () {
            if ($("#name").val()) {
               $("#namediv").attr("class", "form-group has-success has-feedback");
               $("#nameicon").attr("class", "glyphicon glyphicon-ok
               form-control-feedback");
               $("#namesr").text("(success)");
               flag++;
            }
         });
         $("#age").bind("keyup", function () {
            if ($("#age").val() >= 20 && $("#age").val() <= 65) {
               $("#agediv").attr("class", "form-group has-success has-feedback");
               $("#ageicon").attr("class", "glyphicon glyphicon-ok
               form-control-feedback");
               $("#agesr").text("(success)");
               flag++;
            }
         });
         $(":radio").bind("click", function () {
            $("#genderdiv").attr("class", "form-group has-success has-feedback");
            flag++;
         });
         $("#city").bind("change", function () {
            if ($("#city").val()) {
               $("#citydiv").attr("class", "form-group has-success has-feedback");
               flag++;
               if (flag == 6) {
                  $('button').removeAttr("disabled");
               }
            }
```

```
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
                    form-control-feedback"
3894                    aria-hidden="true"></span>
3895                  <span id="useridsr" class="sr-only">(error)</span>
3896                </div>
3897              </div>
3898              <div id="passwddiv" class="form-group has-error has-feedback">
3899                <label for="passwd" class="col-sm-3 control-label">Password</label>
3900                <div class="col-sm-6">
3901                  <div class="input-group">
3902                    <span class="input-group-addon"><span class="glyphicon
3903                      glyphicon-lock"></span></span>
3904                    <input type="password" name="passwd" id="passwd"
                    class="form-control"
3905                    placeholder="Enter your password">
3906                  </div>
3907                  <span id="passwdicon" class="glyphicon glyphicon-remove
                    form-control-feedback"
3908                    aria-hidden="true"></span>
3909                  <span id="passwdsr" class="sr-only">(error)</span>
3910                </div>
3911              </div>
3912              <div id="namediv" class="form-group has-error has-feedback">
3913                <label for="name" class="col-sm-3 control-label">Name</label>
3914                <div class="col-sm-6">
3915                  <div class="input-group">
3916                    <span class="input-group-addon"><span class="glyphicon
3917      glyphicon-user"></span></span>
3918                    <input type="text" name="name" id="name" class="form-control"
                    placeholder="Enter your name">
3919                  </div>
3920                  <span id="nameicon" class="glyphicon glyphicon-remove
                    form-control-feedback"
3921                    aria-hidden="true"></span>
3922                  <span id="namesr" class="sr-only">(error)</span>
3923                </div>
3924              </div>
3925              <div id="agediv" class="form-group has-error has-feedback">
```

```
                    <label for="age" class="col-sm-3 control-label">Age</label>
                    <div class="col-sm-4">
                       <div class="input-group">
                          <span class="input-group-addon"><span class="glyphicon
        glyphicon-music"></span></span>
                          <input type="number" name="age" id="age" class="form-control"
                          placeholder="Enter your age">
                       </div>
                       <span id="ageicon" class="glyphicon glyphicon-remove
                       form-control-feedback"
                          aria-hidden="true"></span>
                       <span id="agesr" class="sr-only">(error)</span>
                    </div>
                 </div>
                 <div id="genderdiv" class="form-group has-error has-feedback">
                    <label class="col-sm-3 control-label">Gender</label>
                    <div class="col-sm-6">
                       <label class="radio-inline">
                          <input type="radio" name="gender" value="1">Male
                       </label>
                       <label class="radio-inline">
                          <input type="radio" name="gender" value="0">Female
                       </label>
                    </div>
                 </div>
                 <div id="citydiv" class="form-group has-error has-feedback">
                    <label for="city" class="col-sm-3 control-label">City</label>
                    <div class="col-sm-6">
                       <select name="city" id="city" class="form-control">
                          <option value="">--선택--</option>
                          <option value="서울">서울</option>
                          <option value="부산">부산</option>
                          <option value="대전">대전</option>
                          <option value="광주">광주</option>
                          <option value="대구">대구</option>
                       </select>
                    </div>
                 </div>
                 <div class="form-group">
                    <div class="col-sm-6 col-sm-offset-3">
                       <button class="col-sm-4 btn btn-primary" disabled>Submit</button>
                    </div>
                 </div>
              </form>
           </div>
        </div>
     </body>
     </html>

4)MainController.java 수정

     @Controller
     public class MainController {

        @Autowired
        private MemberService memberService;

        @GetMapping("/")
```

```
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) {
```

```
4040          this.sqlSession.insert("insert", member);
4041        }
4042
4043    7)mybatis-mapper.xml 수정하기
4044
4045      <parameterMap type="memberVO" id="insertParameterMap">
4046        <parameter property="userid" javaType="java.lang.String" jdbcType="VARCHAR"
          mode="IN"/>
4047        <parameter property="passwd" javaType="java.lang.String" jdbcType="VARCHAR"
          mode="IN"/>
4048        <parameter property="name" javaType="java.lang.String" jdbcType="VARCHAR"
          mode="IN"/>
4049        <parameter property="age" javaType="java.lang.Integer" jdbcType="INTEGER"
          mode="IN"/>
4050        <parameter property="gender" javaType="java.lang.String" jdbcType="VARCHAR"
          mode="IN"/>
4051        <parameter property="city" javaType="java.lang.String" jdbcType="VARCHAR"
          mode="IN"/>
4052      </parameterMap>
4053      <insert id="insert" parameterType="memberVO" parameterMap="insertParameterMap"
          statementType="CALLABLE">
4054        { call member_insert_sp(?,?,?,?,?,?) }
4055      </insert>
4056
4057    8)member_insert_sp
4058
4059      CREATE OR REPLACE PROCEDURE member_insert_sp
4060      (
4061        t_userid          IN          member.userid%TYPE,
4062        t_passwd          IN          member.passwd%TYPE,
4063        t_name            IN          member.name%TYPE,
4064        t_age             IN          member.age%TYPE,
4065        t_gender          IN          member.gender%TYPE,
4066        t_city            IN          member.city%TYPE
4067      )
4068      IS
4069      BEGIN
4070        INSERT INTO Member(userid, passwd, name, age, gender, city)
4071        VALUES(t_userid, t_passwd, t_name, t_age, t_gender, t_city);
4072        COMMIT;
4073      END;
4074
4075    9)http://localhost:8080/ -> 5명 정도 회원가입 진행
4076
4077
4078  17. 모든 회원정보 가져오기
4079    1)index.html 코드 수정하기
4080
4081      <a class="btn btn-primary btn-lg" th:href="@{register}" role="button">회원가입
          </a>  
4082      <a class="btn btn-success btn-lg" th:href="@{list}" role="button">전체회원명단</a>
4083
4084    2)MainController.java 수정하기
4085
4086      @PostMapping("/register")
4087      public String register(MemberVO member) {
4088        //log.info("member = {}", member);
4089        this.memberService.insertMember(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="@{/}"><img th:src="@{images/spring-boot.png}" /></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>
4179
4180    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"
          mode="OUT" resultMap="selectMap"/>
4193      </parameterMap>
4194
4195      <select id="selectAll" parameterMap="selectAllParameterMap"
          statementType="CALLABLE">
4196        { call member_select_all_sp(?) }
4197      </select>
4198
4199    7)member_select_all_sp
4200
4201      CREATE OR REPLACE PROCEDURE member_select_all_sp
4202      (
4203        member_record    OUT    SYS_REFCURSOR
```

```
    )
    AS
    BEGIN
      OPEN member_record FOR
      SELECT * FROM Member
      ORDER BY userid ASC;
    END;
```

8)http://localhost:8080/list

18. 특정 회원 정보 가져오기

1)list.html 수정하기

```
    <tr th:each="member : ${members}">
      <td><a th:href="@{/member/{userid}(userid=${member.userid})}"
              th:text="${member.userid}"></a></td>
      <td th:text="${member.name}"></td>
      <td th:text="${member.age}"></td>
      <td th:if="${member.gender eq '1'}" th:text="남성"></td>
      <td th:if="${member.gender eq '0'}" th:text="여성"></td>
      <td th:text="${member.city}"></td>
    </tr>
```

2)MainController.java

```
    @GetMapping("/member/{userid}")
    public String display(@PathVariable String userid, Model model) {
      Map<String, Object> map = new HashMap<String, Object>();
      map.put("userid", userid);
      this.memberService.selectMember(map);
      List<MemberVO> list = (List<MemberVO>)map.get("result");
      MemberVO member = list.get(0);
      model.addAttribute("member", member);
      //log.info("user = {}", member);
      return "display";   //templates/display.html
    }
```

3)MemberServiceImpl.java

```
    @Override
    public void selectMember(Map map) {
      this.memberDao.read(map);
    }
```

4)MemberDaoImpl.java

```
    @Override
    public void read(Map map) {
      this.sqlSession.selectOne("select", map);
    }
```

5)mybatis-mapper.xml

```
    <parameterMap type="hashmap" id="selectParameterMap">
      <parameter property="userid" javaType="java.lang.String" jdbcType="VARCHAR"
      mode="IN"/>
      <parameter property="result" javaType="ResultSet" jdbcType="CURSOR" mode="OUT"
```

```
      resultMap="selectMap"/>
    </parameterMap>
    <select id="select" parameterMap="selectParameterMap" statementType="CALLABLE">
      { call member_select_sp(?,?) }
    </select>

  6)member_select_sp

    CREATE OR REPLACE PROCEDURE member_select_sp
    (
      t_userid        IN     member.userid%TYPE,
      member_record    OUT    SYS_REFCURSOR
    )
    AS
    BEGIN
      OPEN member_record FOR
      SELECT * FROM Member
      WHERE userid = t_userid;
    END;

  7)templates/display.html

    <!DOCTYPE html>
    <html lang="en" xmlns:th="https://www.thymeleaf.org/">
    <head>
       <meta charset="UTF-8" />
       <meta name="viewport" content="width=device-width, initial-scale=1.0" />
       <title>회원정보 상세보기</title>
    </head>
    <body>
      <div><a href="#" th:href="@{/}"><img th:src="@{/images/spring-boot.png}"
      /></a></div>
      <div th:object="${member}">
        <h1><span th:text="*{name}"></span>(<span th:text="*{userid}"></span>)님!
        회원 정보</h1>
        <ul>
          <li>아이디 : <span th:text="*{userid}"></span></li>
          <li>이름 : <span th:text="*{name}"></span></li>
          <li>나이 : <span th:text="*{age}"></span></li>
          <li>성별 : <span th:if="*{gender eq '1'}" th:text="남성"></span>
                <span th:if="*{gender eq '0'}" th:text="여성"></span>
          </li>
          <li>거주지 : <span th:text="*{city}"></span></li>
        </ul>
       </div>
       <div>
         <a href="#" th:href="@{/list}">목록으로</a>
       </div>
    </body>
    </html>

  8)list에서 특정 회원 click


19. 회원정보 수정하기
  1)display.html 수정하기

    <a href="#" th:href="@{/list}">목록으로</a>  
```

```
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
```

```
    )
    IS
    BEGIN
      UPDATE Member SET age = t_age, gender = t_gender, city = t_city
      WHERE userid = t_userid;
      COMMIT;
    END;
```

7)templates/update.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="https://www.thymeleaf.org/">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>회원정보 수정창</title>
    <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
    <script th:src="@{/js/jquery-3.5.0.min.js}"></script>
    <script>
        $(function () {
            //var flag = 0;
            $("#userid").bind("change", function () {
                if ($("#userid").val()) {
                    $("#useriddiv").attr("class", "form-group has-success has-feedback");
                    $("#useridicon").attr("class", "glyphicon glyphicon-ok
                    form-control-feedback");
                    $("#useridsr").text("(success)");
                    //flag++;
                }
            });
            $("#passwd").bind("change", function () {
                if ($("#passwd").val()) {
                    $("#passwddiv").attr("class", "form-group has-success has-feedback");
                    $("#passwdicon").attr("class", "glyphicon glyphicon-ok
                    form-control-feedback");
                    $("#passwdsr").text("(success)");
                    //flag++;
                }
            });
            $("#name").bind("change", function () {
                if ($("#name").val()) {
                    $("#namediv").attr("class", "form-group has-success has-feedback");
                    $("#nameicon").attr("class", "glyphicon glyphicon-ok
                    form-control-feedback");
                    $("#namesr").text("(success)");
                    //flag++;
                }
            });
            $("#age").bind("keyup", function () {
                if ($("#age").val() >= 20 && $("#age").val() <= 65) {
                    $("#agediv").attr("class", "form-group has-success has-feedback");
                    $("#ageicon").attr("class", "glyphicon glyphicon-ok
                    form-control-feedback");
                    $("#agesr").text("(success)");
                    //flag++;
                }
            });
            $(":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>
4456                         <input type="text" name="userid" id="userid" class="form-control"
4457                         placeholder="Enter your ID" readonly="readonly"
                           th:attrappend="value=*{userid}">
4458                      </div>
4459                      <span id="useridicon" class="glyphicon glyphicon-remove
                        form-control-feedback"
4460                         aria-hidden="true"></span>
4461                      <span id="useridsr" class="sr-only">(error)</span>
4462                   </div>
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>
4470                         <input type="password" name="passwd" id="passwd"
                           class="form-control"
4471                         placeholder="Enter your password">
4472                      </div>
4473                      <span id="passwdicon" class="glyphicon glyphicon-remove
                        form-control-feedback"
4474                         aria-hidden="true"></span>
4475                      <span id="passwdsr" class="sr-only">(error)</span>
4476                   </div>
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"
                            th:attrappend="value=*{name}">
4486                    </div>
4487                    <span id="nameicon" class="glyphicon glyphicon-remove
                        form-control-feedback"
4488                        aria-hidden="true"></span>
4489                    <span id="namesr" class="sr-only">(error)</span>
4490                </div>
4491            </div>
4492            <div id="agediv" class="form-group has-error has-feedback">
4493                <label for="age" class="col-sm-3 control-label">Age</label>
4494                <div class="col-sm-4">
4495                    <div class="input-group">
4496                        <span class="input-group-addon"><span class="glyphicon
4497        glyphicon-music"></span></span>
4498                        <input type="number" name="age" id="age" class="form-control"
4499                        placeholder="Enter your age" th:attrappend="value=*{age}">
4500                    </div>
4501                    <span id="ageicon" class="glyphicon glyphicon-remove
                        form-control-feedback"
4502                        aria-hidden="true"></span>
4503                    <span id="agesr" class="sr-only">(error)</span>
4504                </div>
4505            </div>
4506            <div id="genderdiv" class="form-group has-error has-feedback">
4507                <label class="col-sm-3 control-label">Gender</label>
4508                <div class="col-sm-6">
4509                    <label class="radio-inline">
4510                        <input type="radio" name="gender" value="1"
4511                         th:checked="*{(gender eq '1') ? 'checked' : ''}">Male
4512                    </label>
4513                    <label class="radio-inline">
4514                        <input type="radio" name="gender" value="0"
4515                        th:checked="*{(gender eq '0') ? 'checked' : ''}">Female
4516                    </label>
4517                </div>
4518            </div>
4519            <div id="citydiv" class="form-group has-error has-feedback">
4520                <label for="city" class="col-sm-3 control-label">City</label>
4521                <div class="col-sm-6">
4522                    <select name="city" id="city" class="form-control">
4523                        <option value="">--선택--</option>
4524                        <option value="서울" th:selected="*{(city eq '서울') ? 'selected' :
                        false}">서울</option>
4525                        <option value="부산" th:selected="*{(city eq '부산') ? 'selected' :
                        false}">부산</option>
4526                        <option value="대전" th:selected="*{(city eq '대전') ? 'selected' :
                        false}">대전</option>
4527                        <option value="광주" th:selected="*{(city eq '광주') ? 'selected' :
                        false}">광주</option>
4528                        <option value="대구" th:selected="*{(city eq '대구') ? 'selected' :
                        false}">대구</option>
4529                    </select>
4530                </div>
4531            </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>  
4550    <a th:href="@{/member/update/{userid}(userid=${member.userid})}">정보수정하기
      </a>  
4551    <a th:href="@{/delete/{userid}(userid=${member.userid})}">탈퇴하기</a>
4552
4553  2)MainController.java
4554
4555    @RequestMapping(value="/delete/{userid}", method=RequestMethod.GET)
4556    public String delete(@PathVariable String userid) {
4557      this.memberService.deleteMember(userid);
4558      return "redirect:/";
4559    }
4560
4561  3)MemberServiceImpl.java
4562
4563    @Override
4564    public void deleteMember(String userid) {
4565      this.memberDao.delete(userid);
4566    }
4567
4568  4)MemberDaoImpl.java
4569
4570    @Override
4571    public void delete(String userid) {
4572      this.sqlSession.delete("delete", userid);
4573    }
4574
4575  5)mybatis-mapper.xml
4576
4577    <delete id="delete" parameterType="java.lang.String" statementType="CALLABLE">
4578      { call member_delete_sp(#{userid}) }
4579    </delete>
4580
4581  6)member_delete_sp
4582
4583    CREATE OR REPLACE PROCEDURE member_delete_sp
4584    (
4585      t_userid       IN     member.userid%TYPE
4586    )
4587    IS
4588    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
4691 3)server.xml 변경
4692
4693 <Context docBase="SpringBootRestAPIDemo" path="/SpringBootRestAPIDemo"
reloadable="true" source="org.eclipse.jst.jee.server:SpringBootRestAPIDemo"/>
4694
4695
4696 3. 전체 Member 조회하기
4697 1)MainController.java 수정하기
4698
4699 @Slf4j
4700 @RestController <---변경
4701 public class MainController {
4702 @Autowired
4703 private MemberService memberService;

```
...
...
@RequestMapping(value = "/members", method = RequestMethod.GET)
public Map mebers() {
  Map<String, Object> map = new HashMap<String, Object>();
  this.memberService.selectAllMembers(map);
  List<MemberVO> list = (List<MemberVO>)map.get("results");
  list.forEach(member -> log.info("" + member));
  map.put("code", "success");
  return map;
}
```

2)POSTMAN에서 Test
GET http://localhost:8080/SpringBootRestAPIDemo/members Send

```
{
  "code": "success",
  "results": [
    {
      "userid": "developer",
      "passwd": "12345678",
      "name": "배지민",
      "age": 30,
      "gender": "0",
      "city": "대구"
      },
      {
      "userid": "jimin",
      "passwd": "12345678",
      "name": "한지민",
      "age": 24,
      "gender": "0",
      "city": "대전"
    }
  ]
}
```

3)src/main/resources/static > right-click > New > Other > Web > HTML File
-File name : index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>example.com 회원명단</title>
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link rel="stylesheet" href="css/bootstrap-theme.min.css">
  <style type="text/css">
    body {
      margin: 0px;
      padding: 0px;
    }

    table {
      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 +
                                "'>" + members[i].userid + "</a></td>"
4806                          + "<td>" + members[i].name + "</td>"
4807                          + "<td>" + members[i].age + "</td>"
4808                          + "<td>" + gender + "</td>"
4809                          + "<td>" + members[i].city + "</td>";
4810                        str += "</tr>";
4811                      }
4812                      $("#result").html(str);
4813                    }
4814                  });
4815              });
4816          </script>
4817        </head>
4818
```

```
<body>
  <div>
    <a href="/SpringBootRestAPIDemo"><img src="images/spring-boot.png" /></a>
  </div>
  <h1 class="text-center">회원명단</h1>
  <div style="text-align: center">
    <a href="register.html">Member Add</a>
  </div>
  <table border="1">
    <thead>
      <tr>
        <th>아이디</th>
        <th>이름</th>
        <th>나이</th>
        <th>성별</th>
        <th>거주지</th>
      </tr>
    </thead>
    <tbody id="result">
    </tbody>
  </table>
</body>

</html>
```

4)MainController.java 수정 - 아래 코드 삭제

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

4. 특정 Member 조회하기

1)MainController.java

```
@RequestMapping(value = "/members/{userid}", method = RequestMethod.GET)
public Map display1(@PathVariable String userid) {
  Map<String, Object> map = new HashMap<String, Object>(); map.put("userid",
  userid);
  this.memberService.selectMember(map);
  List<MemberVO> list = (List<MemberVO>)map.get("result");
  MemberVO member = list.get(0);
  log.info("user = {}", member);
  map.remove("userid");
  map.remove("result");
  map.put("result", member);
  map.put("code","success");
  return map;
}
```

2)POSTMAN에서 Test

GET http://localhost:8080/SpringBootRestAPIDemo/members/jimin Send

```
{
  "result": [
    {
```

```
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/"><img src="images/spring-boot.png"
          /></a></div>
4922        <h1>Member Information</h1>
4923        <h1><span class="name"></span>(<span class="userid"></span>)님! 회원 정보</h1>
4924        <ul>
4925          <li>아이디 : <span class="userid"></span></li>
4926          <li>이름 : <span class="name"></span></li>
4927          <li>나이 : <span id="age"></span></li>
4928          <li>성별 : <span id="gender"></span></li>
4929          <li>거주지 : <span id="city"></span></li>
4930        </ul>
4931
4932        <div>
```

```
        <a href="javascript:void(0)" onclick="javascript:history.back();">목록으로</a>>
      </div>
    </body>

    </html>
```

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

5. Member 등록 구현하기

1)MainController.java

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

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

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

```
4991            $("#useridsr").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"
                    placeholder="Enter your ID">
5076                  </div>
5077                  <span id="useridicon" class="glyphicon glyphicon-remove
                    form-control-feedback" aria-hidden="true"></span>
5078                  <span id="useridsr" class="sr-only">(error)</span>
5079                </div>
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"
                      class="form-control" placeholder="Enter your password">
5089                  </div>
5090                  <span id="passwdicon" class="glyphicon glyphicon-remove
                    form-control-feedback" aria-hidden="true"></span>
5091                  <span id="passwdsr" class="sr-only">(error)</span>
5092                </div>
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">
                </div>
                <span id="nameicon"  class="glyphicon glyphicon-remove
                form-control-feedback"  aria-hidden="true"></span>
                <span id="namesr" class="sr-only">(error)</span>
              </div>
            </div>
            <div id="agediv" class="form-group has-error has-feedback">
              <label for="age" class="col-sm-3 control-label">Age</label>
              <div class="col-sm-4">
                <div class="input-group">
                  <span class="input-group-addon">
                    <span class="glyphicon glyphicon-music"></span>
                  </span>
                    <input type="number" name="age" id="age" class="form-control"
                    placeholder="Enter your age">
                </div>
                <span id="ageicon"  class="glyphicon glyphicon-remove
                form-control-feedback"  aria-hidden="true"></span>
                <span id="agesr" class="sr-only">(error)</span>
              </div>
            </div>
            <div id="genderdiv" class="form-group has-error has-feedback">
              <label class="col-sm-3 control-label">Gender</label>
              <div class="col-sm-6">
                <label class="radio-inline">
                  <input type="radio" name="gender" value="1" class="gender">Male
                </label>
                <label class="radio-inline">
                  <input type="radio" name="gender" value="0" class="gender">Female
                </label>
              </div>
            </div>
            <div id="citydiv" class="form-group has-error has-feedback">
              <label for="city" class="col-sm-3 control-label">City</label>
              <div class="col-sm-6">
                <select name="city" id="city" class="form-control">
                  <option value="">--선택--</option>
                  <option value="서울">서울</option>
                  <option value="부산">부산</option>
                  <option value="대전">대전</option>
                  <option value="광주">광주</option>
                  <option value="대구">대구</option>
                </select>
              </div>
            </div>
            <div class="form-group">
              <div class="col-sm-6 col-sm-offset-3">
                <button type="button" class="col-sm-4 btn btn-primary"
                disabled>Submit</button>
              </div>
            </div>
          </form>
        </div>
      </div>
    </body>
    </html>
```

6. Member 정보 수정 구현하기
1)MainController.java

```
@RequestMapping(value = "/members", method = RequestMethod.PUT)
public Map update(@RequestBody MemberVO member) {
  log.info("수정할 멤버 = " + member);
  this.memberService.updateMember(member);
  Map<String, Object> map = new HashMap<String, Object>();
  map.put("code", "success");
  return map;
}
```

2)Postman

PUT http://localhost:8080/SpringBootRestAPIDemo/members
Body > raw > JSON(application/json)

```
{
  "userid" : "girlsage",
  "name" : "소년시대",
  "gender" : "남성",
  "city" : "부산"
}
```

Send 버튼 클릭하면
Body
{"code": "success"}

3)display.html 수정
-다음 code를 추가한다.

```
<a href="javascript:void(0)" onclick="javascript:history.back();">목록으로
</a>  
<a href="javascript:void(0)" id="btnUpdate">수정하기</a>

<script>
  $(function () {
    userid = location.search.substring(1).split("=")[1];
    $.ajax({
      url: "/SpringBootRestAPIDemo/members/" + userid,
      type: "GET",
      dataType : "json",
      success: function (data) {
        var member = data.result;
        $(".userid").text(member.userid);
        $(".name").text(member.name);
        $("#age").text(member.age);
        if(member.gender == '1') $("#gender").text("남성");
        else if(member.gender == '0') $("#gender").text("여성");
        $("#city").text(member.city);
      },
      error : function(err){
        alert('error = ' + err);
      }
    });
    //여기부터 코드 추가
    $("#btnUpdate").bind("click", function(){
      location.href = "update.html?userid=" + userid;
```

```
            });
          });
        </script>

4)templates/update.html

    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>회원정보 수정창</title>
        <link rel="stylesheet" href="css/bootstrap.min.css">
        <script src="js/jquery-3.5.0.min.js"></script>
        <script>
            $(function () {
             userid = location.search.substring(1).split("=")[1];
             $.ajax({
               url: "/SpringBootRestAPIDemo/members/" + userid,
               type: "GET",
               dataType : "json",
               success: function (data) {
                  var member = data.result;
                  $("#userid").val(member.userid);
                  $("#name").val(member.name);
                  $("#age").val(member.age);
                  if(member.gender == '1') $("#male").prop("checked", true);
                  else if(member.gender == '0') $("#female").prop("checked", true);
                  switch(member.city){
                     case "서울" : $("#city option:eq(1)").attr("selected", "selected"); break;
                     case "부산" : $("#city option:eq(2)").attr("selected", "selected"); break;
                     case "대전" : $("#city option:eq(3)").attr("selected", "selected"); break;
                     case "광주" : $("#city option:eq(4)").attr("selected", "selected"); break;
                     case "대구" : $("#city option:eq(5)").attr("selected", "selected"); break;
                  }
               },
               error : function(err){
                  alert('error = ' + err);
               }
             });

             $("#useriddiv").attr("class", "form-group has-success has-feedback");
               $("#useridicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
               $("#useridsr").text("(success)");

               $("#passwddiv").attr("class", "form-group has-success has-feedback");
               $("#passwdicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
               $("#passwdsr").text("(success)");

               $("#namediv").attr("class", "form-group has-success has-feedback");
               $("#nameicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
               $("#namesr").text("(success)");

               $("#agediv").attr("class", "form-group has-success has-feedback");
               $("#ageicon").attr("class", "glyphicon glyphicon-ok form-control-feedback");
               $("#agesr").text("(success)");

               $("#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"
                  placeholder="Enter your ID">
5308                </div>
5309                <span id="useridicon" class="glyphicon glyphicon-remove
                  form-control-feedback" aria-hidden="true"></span>
5310                <span id="useridsr" class="sr-only">(error)</span>
5311              </div>
5312            </div>
5313            <div id="passwddiv" class="form-group has-error has-feedback">
5314              <label for="passwd" class="col-sm-3 control-label">Password</label>
5315              <div class="col-sm-6">
5316                <div class="input-group">
5317                  <span class="input-group-addon">
5318                    <span class="glyphicon glyphicon-lock"></span>
5319                  </span>
5320                  <input type="password" name="passwd" id="passwd"
                  class="form-control" placeholder="Enter your password">
5321                </div>
5322                <span id="passwdicon" class="glyphicon glyphicon-remove
                  form-control-feedback" aria-hidden="true"></span>
5323                <span id="passwdsr" class="sr-only">(error)</span>
```

```
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>
```

```
            <div class="form-group">
              <div class="col-sm-6 col-sm-offset-3">
                <button type="button" class="col-sm-4 btn btn-primary">수정하기</button>
              </div>
            </div>
          </form>
        </div>
      </div>
    </body>
    </html>
```

7. Member 정보 삭제 구현하기

1)MainController.java

```
@RequestMapping(value="/members/{userid}", method=RequestMethod.DELETE)
 public Map delete(@PathVariable String userid) {
    this.memberService.deleteMember(userid);
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("code", "success");
    return map;
 }
```

2)POSTMAN

DELETE http://localhost:8080/SpringBootRestAPIDemo/members/girlsage
Send button click하면
Body
{"code": "success"}

3)display.html 수정

-아래의 code를 추가한다.

```
<a href="javascript:void(0)" onclick="javascript:history.back();">목록으로
</a>  
<a href="javascript:void(0)" id="btnUpdate">수정하기</a>  
<a href="javascript:void(0)" id="btnDelete">탈퇴하기</a>

...
...
$("#btnDelete").bind("click", function(){
  if(confirm("정말 탈퇴하시겠습니까?")){
    $.ajax({
      url : "/SpringBootRestAPIDemo/members/" + userid,
      type : "DELETE",
      success : function(data){
        alert(data.code);
        location.href = "/SpringBootRestAPIDemo/";
      }
    });
  }else{
    location.go(0);
  }
});
```

Task16. Spring Boot와 JPA 연동하기

1. Spring Boot의 Database 처리는 기본적으로 JPA 기술을 기반으로 하고 있다.
2. 이 JPA를 Spring Framework에서 사용할 수 있게 한 것이 'Spring Data JPA' framework이다.
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.
4. 즉, 이 library는 각종 library를 조합해서 간단히 database 접속을 구현하게 한 기능이다.

5. Spring Boot project 생성
   1)Package Explorer > right-click > New > Spring Starter Project
   2)다음 각 항목의 값을 입력한 후 Next 클릭
     -Service URL :http://start.spring.io
     -Name : JpaDemo
     -Type : Maven
     -Packaging : Jar
     -Java Version : 8
     -Language : Java
     -Group : com.example
     -Artifact : JpaDemo
     -Version : 0.0.1-SNAPSHOT
     -Description : Demo project for Spring Boot
     -Package : com.example.biz

   3)다음 각 항목을 선택한 후 Finish 클릭
     -Spring Boot Version : 2.2.4
     -Select
       --SQL > Spring Data JPA, H2 Database
       --Developer Tools > Spring Boot DevTools
     -Finish

6. Entity
   1)JPA에서 Entity라는 것은 Database에 저장하기 위해서 정의한 class이다.
   2)일반적으로 RDBMS에서 Table 같은 것이다.
   3)com.example.biz > right-click > New > Class

   4)Name : MemberVO

```
package com.example.biz;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;


@Entity(name="Member")
public class MemberVO {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private long id;
  @Column
  private String username;
  @Column
  private int age;

  public MemberVO() {}
```

```
5489        public MemberVO(String username, int age) {
5490          this.username = username;
5491          this.age = age;
5492        }
5493
5494        public long getId() {
5495          return id;
5496        }
5497
5498        public void setId(long id) {
5499          this.id = id;
5500        }
5501
5502        public String getUsername() {
5503          return username;
5504        }
5505
5506        public void setUsername(String username) {
5507          this.username = username;
5508        }
5509
5510        public int getAge() {
5511          return age;
5512        }
5513
5514        public void setAge(int age) {
5515          this.age = age;
5516        }
5517
5518        @Override
5519        public String toString() {
5520          return "MemberVO [id=" + id + ", username=" + username + ", age=" + age + "]";
5521        }
5522      }
5523
5524    5)여기서 주의할 점은 기본 생성자는 반드시 넣어야 한다.
5525
5526
5527  7. Repository
5528    1)Entity class를 구성했다면 이번엔 Repository interface를 만들어야 한다.
5529    2)Spring Framework에서는 Entity의 기본적인 삽입, 조회, 수정, 삭제가 가능하도록 CrudRepository라는
        inteface가 있다.
5530    3)com.example.biz > right-click > New > Interface
5531
5532    4)Name : MemberRepository
5533
5534      package com.example.biz;
5535
5536      import java.util.List;
5537
5538      import org.springframework.data.jpa.repository.Query;
5539      import org.springframework.data.repository.CrudRepository;
5540      import org.springframework.data.repository.query.Param;
5541
5542      public interface MemberRepository extends CrudRepository<MemberVO, Long> {
5543        List<MemberVO> findByUsernameAndAgeLessThan(String username, int age);
5544
5545        @Query("select t from Member t where username= :username and age < :age")
```

```
5546        List<MemberVO> findByUsernameAndAgeLessThanSQL(@Param("username") String
            username, @Param("age") int age);
5547
5548        List<MemberVO> findByUsernameAndAgeLessThanOrderByAgeDesc(String username,
            int age);
5549      }
5550
5551      -위의 코드는 실제로 MemberVO Entity를 이용하기 위한 Repository class이다.
5552      -기본적인 method 외에도 추가적인 method를 지정할 수 있다.
5553      -method 이름을 기반(Query Method)으로 해서 만들어도 되고 @Query를 이용해 기존의 SQL처럼 만들어도
          된다.
5554      -findByUsernameAndAgeLessThan method와 findByUsernameAndAgeLessThanSQL method
          는 같은 결과를 출력하지만 전자의 method는 method 이름을 기반으로 한 것이고 후자의 method는 @Query
          annotation을 기반으로 해서 만든 것이다.
5555      -method 이름 기반으로 해서 만들면 추후에 사용할 때 method 이름만으로도 어떤 query인지 알 수 있다는 장
          점이 있다.
5556      -반대로 @Query annotation으로 만든 method는 기존의 source를 converting하는 경우 유용하게 사용할
          수 있다.
5557      -@Query
5558        --다만 @Query annotation으로 query를 만들 때에 from 절에 들어가는 table은 Entity로 지정된 class
            이름이다.
5559      -method 이름 기반 작성법
5560      -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.
5561
5562
5563  8. Application 작성
5564    1)Entity 및 Repository가 준비 되었다면 실제로 @SpringBootApplication annotation이 있는 class에서
        실제로 사용해 보자.
5565
5566      package com.example.biz;
5567
5568      import java.util.List;
5569
5570      import org.springframework.beans.factory.annotation.Autowired;
5571      import org.springframework.boot.CommandLineRunner;
5572      import org.springframework.boot.SpringApplication;
5573      import org.springframework.boot.autoconfigure.SpringBootApplication;
5574
5575      @SpringBootApplication
5576      public class JpaDemoApplication implements CommandLineRunner {
5577        @Autowired
5578        MemberRepository memberRepository;
5579
5580        public static void main(String[] args) {
5581          SpringApplication.run(JpaDemoApplication.class, args);
5582        }
5583
5584        @Override
5585        public void run(String... args) throws Exception {
5586          memberRepository.save(new MemberVO("a", 10));
5587          memberRepository.save(new MemberVO("b", 15));
5588          memberRepository.save(new MemberVO("c", 10));
5589          memberRepository.save(new MemberVO("a", 5));
5590          Iterable<MemberVO> list1 = memberRepository.findAll();
5591          System.out.println("findAll() Method.");
5592          for (MemberVO m : list1) {
```

```
            System.out.println(m.toString());
          }
          System.out.println("findByUserNameAndAgeLessThan() Method.");
          List<MemberVO> list2 = memberRepository.findByUsernameAndAgeLessThan("a",
          10);
          for (MemberVO m : list2) {
            System.out.println(m.toString());
          }
          System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
          List<MemberVO> list3 =
          memberRepository.findByUsernameAndAgeLessThanSQL("a", 10);
          for (MemberVO m : list3) {
            System.out.println(m.toString());
          }

          System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
          List<MemberVO> list4 =
          memberRepository.findByUsernameAndAgeLessThanOrderByAgeDesc("a", 15);
          for (MemberVO m : list4) {
            System.out.println(m.toString());
          }
          memberRepository.deleteAll();
        }

      }


9. 실행

  findAll() Method.
  MemberVO [id=1, username=a, age=10]
  MemberVO [id=2, username=b, age=15]
  MemberVO [id=3, username=c, age=10]
  MemberVO [id=4, username=a, age=5]
  findByUserNameAndAgeLessThan() Method.
  MemberVO [id=4, username=a, age=5]
  findByUserNameAndAgeLessThanSQL() Method.
  MemberVO [id=4, username=a, age=5]
  findByUserNameAndAgeLessThanSQL() Method.
  MemberVO [id=1, username=a, age=10]
  MemberVO [id=4, username=a, age=5]



-----------------------------------------------------
Task16. Spring Boot JPA
  1. Spring Boot project 생성
    1)Package Explorer > right-click > New > Spring Starter Project

    2)다음의 각 항목의 값을 입력 후 Next 클릭
      -Service URL :http://start.spring.io
      -Name : BootJpaDemo
      -Type : Maven
      -Packaging : Jar
      -Java Version : 8
      -Language : Java
      -Group : com.example
      -Artifact : BootJpaDemo
```

```
5648        -Version : 0.0.1-SNAPSHOT
5649        -Description : Demo project for Spring Boot
5650        -Package : com.example.biz
5651
5652      3)각 항목을 선택 후 Finish 클릭
5653        -Spring Boot Version : 2.2.4
5654        -Select
5655          --SQL > Spring Data JPA, H2 Database
5656          --Developer Tools > Spring Boot DevTools
5657          --Web > Spring Web
5658          --Template Engines > Thymeleaf
5659        -Finish
5660
5661      4)DevTools
5662        -It provides developer tools.
5663        -These tools are helpful in application development mode.
5664        -One of the features of developer tool is automatic restart of the server for any change
            in code
5665
5666
5667  2. Entity 작성하기
5668    1)com.example.biz > right-click > New > Package
5669    2)Name : com.example.biz.vo
5670    3)com.example.biz.vo > right-click > New > Class
5671    4)Name : User
5672
5673      package com.example.biz.vo;
5674
5675      import javax.persistence.Column;
5676      import javax.persistence.Entity;
5677      import javax.persistence.GeneratedValue;
5678      import javax.persistence.GenerationType;
5679      import javax.persistence.Id;
5680      import javax.persistence.Table;
5681
5682      import lombok.Data;
5683
5684      @Entity
5685      @Table
5686      public class User {
5687
5688        @Id
5689        @GeneratedValue(strategy = GenerationType.AUTO)
5690        @Column
5691        private long id;
5692
5693        @Column(length = 20, nullable = false)
5694        private String username;
5695
5696        @Column(length = 100, nullable = true)
5697        private String email;
5698
5699        @Column(nullable = true)
5700        private Integer age;
5701
5702        @Column(nullable = true)
5703        private String memo;
5704
```

```
5705        public long getId() {
5706          return id;
5707        }
5708
5709        public void setId(long id) {
5710          this.id = id;
5711        }
5712
5713        public String getUsername() {
5714          return username;
5715        }
5716
5717        public void setUsername(String username) {
5718          this.username = username;
5719        }
5720
5721        public String getEmail() {
5722          return email;
5723        }
5724
5725        public void setEmail(String email) {
5726          this.email = email;
5727        }
5728
5729        public Integer getAge() {
5730          return age;
5731        }
5732
5733        public void setAge(Integer age) {
5734          this.age = age;
5735        }
5736
5737        public String getMemo() {
5738          return memo;
5739        }
5740
5741        public void setMemo(String memo) {
5742          this.memo = memo;
5743        }
5744      }
5745
5746
5747  3. Repository 생성하기
5748    1)com.example.biz > right-click > New > Package
5749    2)Name : com.example.biz.dao
5750    3)com.example.biz.dao > right-click > New > Interface
5751    4)Name : UserRepository > Finish
5752
5753      package com.example.biz.dao;
5754
5755      import org.springframework.data.jpa.repository.JpaRepository;
5756      import org.springframework.stereotype.Repository;
5757
5758      import com.example.biz.vo.User;
5759
5760      @Repository("repository")
5761      public interface UserRepository extends JpaRepository<User, Long>{
5762
```

```
    }

    -JpaRepository라는 interface는 새로운 repository를 생성하기 위한 토대가 된다.
    -모든 Repository는 이 JpaRepository를 상속해서 작성한다


4. HelloController 작성
  1)com.example.biz > right-click > New > Class
  2)Name : HelloController

    package com.example.biz;

    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.stereotype.Controller;
    import org.springframework.web.bind.annotation.RequestMapping;
    import org.springframework.web.servlet.ModelAndView;

    import com.example.biz.dao.UserRepository;
    import com.example.biz.vo.User;

    @Controller
    public class HelloController {

      @Autowired
      UserRepository repository;

      @RequestMapping("/")
      public ModelAndView index(ModelAndView mav) {
        mav.setViewName("index");
        mav.addObject("msg", "this is sample content.");
        Iterable<User> list = repository.findAll();
        mav.addObject("data", list);
        return mav;
      }
    }

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


5. JUnit Test
  1)src/test/java/com.example.biz.BootJpaDemoApplicationTests.java > right-click > Run As
    > JUnit Test
  2)Green bar


6. template 준비하기
  1)src/main/resources > right-click > New > File

  2)File name : messages.properties
    content.title=Message sample page.
    content.message=This is sample message from properties.

  3)src/main/resources/templates > right-click > New > Web > HTML Files
  4)Name : index.html

    <!DOCTYPE HTML>
```

```
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>top page</title>
    <meta charset="UTF-8" />
    <style>
      h1 { font-size:18pt; font-weight:bold; color:gray; }
      body { font-size:13pt; color:gray; margin:5px 25px; }
      pre { border: solid 3px #ddd; padding: 10px; }
    </style>
  </head>
  <body>
    <h1 th:text="#{content.title}">Hello page</h1>
    <pre th:text="${data}"></pre>
  </body>
</html>
```

7. 실행해서 접속
   1)저장돼있는 data가 회색 사각 틀 안에 표시된다.
   2)아직 아무 data가 없기 때문에 빈 배열 []라고 표시된다.

8. Entity의 CRUD 처리하기 : form으로 data 저장하기
   1)index.html 수정

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>top page</title>
<meta charset="UTF-8" />
<style>
h1 {
  font-size: 18pt;
  font-weight: bold;
  color: gray;
}

body {
  font-size: 13pt;
  color: gray;
  margin: 5px 25px;
}

tr {
  margin: 5px;
}

th {
  padding: 5px;
  color: white;
  background: darkgray;
}

td {
  padding: 5px;
  color: black;
  background: #e0e0ff;
}
```

```
5878      </style>
5879      </head>
5880      <body>
5881        <h1 th:text="#{content.title}">Hello page</h1>
5882        <table>
5883          <form method="post" action="/" th:object="${formModel}">
5884            <tr>
5885              <td><label for="username">이름</label></td>
5886              <td><input type="text" name="username" th:value="*{username}" /></td>
5887            </tr>
5888            <tr>
5889              <td><label for="age">연령</label></td>
5890              <td><input type="text" name="age" th:value="*{age}" /></td>
5891            </tr>
5892            <tr>
5893              <td><label for="email">메일</label></td>
5894              <td><input type="text" name="email" th:value="*{email}" /></td>
5895            </tr>
5896            <tr>
5897              <td><label for="memo">메모</label></td>
5898              <td><textarea name="memo" th:text="*{memo}" cols="20"
                  rows="5"></textarea></td>
5899            </tr>
5900            <tr>
5901              <td></td>
5902              <td><input type="submit" /></td>
5903            </tr>
5904          </form>
5905        </table>
5906        <hr />
5907        <table>
5908          <tr>
5909            <th>ID</th>
5910            <th>이름</th>
5911          </tr>
5912          <tr th:each="obj : ${datalist}">
5913            <td th:text="${obj.id}"></td>
5914            <td th:text="${obj.username}"></td>
5915          </tr>
5916        </table>
5917      </body>
5918      </html
5919
5920    2)HelloController.java 수정
5921
5922      package com.example.biz;
5923
5924      import org.springframework.beans.factory.annotation.Autowired;
5925      import org.springframework.stereotype.Controller;
5926      import org.springframework.transaction.annotation.Transactional;
5927      import org.springframework.web.bind.annotation.ModelAttribute;
5928      import org.springframework.web.bind.annotation.RequestMapping;
5929      import org.springframework.web.bind.annotation.RequestMethod;
5930      import org.springframework.web.servlet.ModelAndView;
5931
5932      import com.example.biz.dao.UserRepository;
5933      import com.example.biz.vo.User;
5934
```

```
5935      @Controller
5936      public class HelloController {
5937
5938        @Autowired
5939        UserRepository repository;
5940
5941        @RequestMapping(value = "/", method = RequestMethod.GET)
5942        public ModelAndView index(@ModelAttribute("formModel") User mydata, ModelAndView
          mav) {
5943          mav.setViewName("index");
5944          mav.addObject("msg", "this is sample content.");
5945          Iterable<User> list = repository.findAll();
5946          mav.addObject("datalist", list);
5947          return mav;
5948        }
5949
5950        @RequestMapping(value = "/", method = RequestMethod.POST)
5951        @Transactional(readOnly = false)
5952        public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
          mav) {
5953          repository.saveAndFlush(mydata);
5954          return new ModelAndView("redirect:/");
5955        }
5956      }
5957
5958
5959  9. @ModelAttribute와 data 저장
5960    1)@ModelAttribute
5961      -이것은 entity class의 instance를 자동으로 적용할 때 사용
5962      -인수에는 instance 이름을 지정한다.
5963      -이것은 전송 form에서 th:object로 지정하는 값이 된다.
5964      -전송된 form의 값이 자동으로 User instance로 저장된다.
5965      -따라서 이 annotation을 이용하면 이렇게 쉽게 전송한 data를 저장할 수 있다.
5966
5967    2)saveAndFlush() method
5968      -HomeController.java의 아래 code를 보자.
5969        @RequestMapping(value = "/", method = RequestMethod.POST)
5970        @Transactional(readOnly=false)
5971        public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
          mav) {
5972          repository.saveAndFlush(mydata);
5973          return new ModelAndView("redirect:/");
5974        }
5975
5976    3)미리 설정한 entity는 JpaRepository의 saveAndFlush라는 method를 통해 entity를 영구화한다.
5977    4)Database를 사용하고 있다면 Database에 그대로 저장된다
5978
5979
5980  10. @Transactional과 transaction
5981    1)바로 위의 code에서 @Transactional(readOnly=false)가 있다.
5982    2)이 annotation은 transaction을 위한 것이다.
5983    3)이 annotation때문에 method내에서 실행되는 database 처리가 일괄적으로 실행되게 된다.
5984    4)data 변경 처리는 도중에 외부 접속에 의해 data 구조나 내용이 바뀌면 data 일관성에 문제가 발생하게 된다.
5985    5)이런 문제를 방지하기 위해 transaction이 사용되는 것이다
5986    6)code를 보면 readOnly=false라고 설정하고 있다.
5987    7)이 readOnly는 문자 그대로 '읽기 전용(변경 불가)임을 의미한다.
5988    8)readOnly=false라고 설정하면 변경을 허가하는 transaction이다.
5989
```

11. Data 초기화 처리
1)저장한 data는 application을 종료하고 다시 실행하면 지워진다.
2)HSQLDB는 기본적으로 memory내에 data를 cache하고 있으므로 종료와 함께 지워지는 것이다.
3)controller에 data를 작성하는 초기화 처리를 넣기로 한다.
4)HelloController.java code 추가

```
@PostConstruct
public void init(){
  User user1 = new User();
  user1.setUsername("한지민");
  user1.setAge(24);
  user1.setEmail("javaexpert@nate.com");
  user1.setMemo("Hello, Spring JPA");
  repository.saveAndFlush(user1);

  User user2 = new User();
  user2.setUsername("조용필");
  user2.setAge(66);
  user2.setEmail("aaa@aaa.com");
  user2.setMemo("Good Morning!");
  repository.saveAndFlush(user2);

  User user3 = new User();
  user3.setUsername("이미자");
  user3.setAge(70);
  user3.setEmail("bbb@bbb.com");
  user3.setMemo("Spring Boot is very good.");
  repository.saveAndFlush(user3);
}
```

5)@PostConstruct는 생성자를 통해 instance가 생성된 후에 호출되는 method임을 나타낸다.
6)Controller는 처음에 한 번만 instance를 만들고 이후에는 해당 instance를 유지한다.
7)따라서 여기에 test용 data 작성 처리를 해두면 application 실행시에 반드시 한 번 실행되어, data가 준비되는 것이다.

12. User Find 및 Update 처리하기
1)src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
2)File name : edit.html > Finish

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>edit page</title>
<style>
h1 {
  font-size: 18pt;
  font-weight: bold;
  color: gray;
}

body {
  font-size: 13pt;
  color: gray;
  margin: 5px 25px;
}
```

```
6047
6048      tr {
6049        margin: 5px;
6050      }
6051
6052      th {
6053        padding: 5px;
6054        color: white;
6055        background: darkgray;
6056      }
6057
6058      td {
6059        padding: 5px;
6060        color: black;
6061        background: #e0e0ff;
6062      }
6063      </style>
6064      </head>
6065      <body>
6066        <h1 th:text="${title}">Edit page</h1>
6067        <table>
6068          <form method="post" action="/edit" th:object="${formModel}">
6069            <input type="hidden" name="id" th:value="*{id}" />
6070            <tr>
6071              <td><label for="username">이름</label></td>
6072              <td><input type="text" name="username" th:value="*{username}" /></td>
6073            </tr>
6074            <tr>
6075              <td><label for="age">연령</label></td>
6076              <td><input type="text" name="age" th:value="*{age}" /></td>
6077            </tr>
6078            <tr>
6079              <td><label for="email">메일</label></td>
6080              <td><input type="text" name="email" th:value="*{email}" /></td>
6081            </tr>
6082            <tr>
6083              <td><label for="memo">메모</label></td>
6084              <td><textarea name="memo" th:text="*{memo}" cols="20"
                  rows="5"></textarea></td>
6085            </tr>
6086            <tr>
6087              <td></td>
6088              <td><input type="submit" /></td>
6089            </tr>
6090          </form>
6091        </table>
6092      </body>
6093      </html>
6094
6095    3)UserRepository code 추가
6096
6097      package com.example.biz.dao;
6098
6099      import java.util.Optional;
6100
6101      import org.springframework.data.jpa.repository.JpaRepository;
6102      import org.springframework.stereotype.Repository;
6103
```

```
6104    import com.example.biz.vo.User;
6105
6106    @Repository("repository")
6107    public interface UserRepository extends JpaRepository<User, Long>{
6108      public Optional<User> findById(Long id);
6109    }
6110
6111  4)RequestHandler 작성하기
6112    -HelloController.java code 추가
6113
6114      @RequestMapping(value = "/edit/{id}", method = RequestMethod.GET)
6115      public ModelAndView edit(@ModelAttribute User user, @PathVariable int id,
        ModelAndView mav) {
6116        mav.setViewName("edit");
6117        mav.addObject("title", "edit mydata.");
6118        Optional<User> findUser = repository.findById((long) id);
6119        mav.addObject("formModel", findUser.get());
6120        return mav;
6121      }
6122
6123      @RequestMapping(value = "/edit", method = RequestMethod.POST)
6124      @Transactional(readOnly = false)
6125      public ModelAndView update(@ModelAttribute User user, ModelAndView mav) {
6126        repository.saveAndFlush(user);
6127        return new ModelAndView("redirect:/");
6128      }
6129
6130  5)접속해서 아래와 같이 URL을 입력하면
6131
6132    http://localhost:8080/edit/1
6133
6134    -해당 ID의 data가 표시된다.
6135    -data를 변경하고 전송해보자.
6136    -findById는 어디서 구현되는 것일까?
6137      --repository는 method의 이름을 기준으로 entity 검색 처리를 자동 생성한다.
6138      --즉 repository에 method 선언만 작성하고, 구체적인 처리를 구현할 필요가 없다.
6139
6140
6141 13. Entity delete 구현하기
6142   1)update를 하고 select를 했으니 이번에는 delete를 해 보자.
6143   2)delete.html template를 작성한다.
6144   3)src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
6145   4)File name : delete.html > Finish
6146
6147    <!DOCTYPE html>
6148    <html xmlns:th="http://www.thymeleaf.org">
6149    <head>
6150    <meta charset="UTF-8">
6151    <title>delete page</title>
6152    <style>
6153    h1 {
6154      font-size: 18pt;
6155      font-weight: bold;
6156      color: gray;
6157    }
6158
6159    body {
6160      font-size: 13pt;
```

```
6161        color: gray;
6162        margin: 5px 25px;
6163      }
6164
6165      td {
6166        padding: 0px 20px;
6167        background: #eee;
6168      }
6169      </style>
6170      </head>
6171      <body>
6172        <h1 th:text="${title}">Delete page</h1>
6173        <table>
6174          <form method="post" action="/delete" th:object="${formModel}">
6175            <input type="hidden" name="id" th:value="*{id}" />
6176            <tr>
6177              <td><p th:text="|이름 :  *{username}|"></p></td>
6178            </tr>
6179            <tr>
6180              <td><p th:text="|연령 :  *{age}|"></p></td>
6181            </tr>
6182            <tr>
6183              <td><p th:text="*{email}"></p></td>
6184            </tr>
6185            <tr>
6186              <td><p th:text="*{memo}"></p></td>
6187            </tr>
6188            <tr>
6189              <td><input type="submit" value="delete" /></td>
6190            </tr>
6191          </form>
6192        </table>
6193      </body>
6194      </html>
6195
6196    5)RequestHandler 작성
6197
6198      @RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
6199        public ModelAndView delete(@PathVariable int id, ModelAndView mav) {
6200          mav.setViewName("delete");
6201          mav.addObject("title", "delete mydata.");
6202          Optional<User> user = repository.findById((long) id);
6203          mav.addObject("formModel", user.get());
6204          return mav;
6205        }
6206
6207        @RequestMapping(value = "/delete", method = RequestMethod.POST)
6208        @Transactional(readOnly = false)
6209        public ModelAndView remove(@RequestParam long id, ModelAndView mav) {
6210          repository.deleteById(id);
6211          return new ModelAndView("redirect:/");
6212        }
6213
6214    6)접속해서 실행
6215    http://localhost:8080/delete/2라고 하면 id가 2번이 출력되고 delete button을 누르면 삭제된다.
```