```
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
  <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>
    <packaging>jar</packaging>

    <name>springbootdemo</name>
    <url>http://maven.apache.org</url>
    <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>2.2.4.RELEASE</version>
    </parent>

    <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <java.version>1.8</java.version>
    </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>
  </project>
```

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 > Project Facets 수정하기

1)Java --> 1.8

2)Runtimes Tab
- Apache Tomcat v9.0 check
- jdk1.8.0_241 check

3)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!";
  }

  public static void main( String[] args ){
```

```
        SpringApplication.run(App.class, args);
      }
    }
```

2)@RestController
-이 annotation을 붙이면 web appication에서 request을 받아들이는 controller class임을 나타낸다.

3)@EnableAutoConfiguration
-이 annotation은 매우 중요하다.
-이 annotation을 붙이면 다양한 설정이 자동으로 수행되고 기존의 spring application에 필요했던 설정 file들이 필요없게 된다.

4)@RequestMapping("/")
-이 annotation이 붙으면 이 method가 HTTP 요청을 받아들이는 method임을 나타낸다.
-@GetMapping도 가능
-@RequestMapping(value="/", method=RequestMethod.GET)과 @GetMapping은 동일하다.

5)return "Hello World!";
-HTTP 응답을 반환한다.
-@RestController annotation이 붙은 class에 속한 method에서 문자열을 반환하면 해당 문자열이 그대로 HTTP 응답이 되어 출력된다.

6)SpringApplication.run(App.clas, args);
-spring boot applicaton을 실행하는 데 필요한 처리를 main() 안에서 작성한다.
-@EnableAutoConfiguration annotation이 붙은 class를 SpringApplication.run()의 첫번째 인자로 지정한다.

6. Web Application 실행하기

1)springbootdemo project > right-click > Run As > Spring Boot App

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

2020-02-18 22:58:42.425  INFO 6624 --- [           main] com.example.springbootdemo.App           : Starting App on DESKTOP-1BKHISM with PID 6624 (C:\SpringHome\springbootdemo\target\classes started by devex in C:\SpringHome\springbootdemo)
2020-02-18 22:58:42.434  INFO 6624 --- [           main] com.example.springbootdemo.App           : No active profile set, falling back to default profiles: default
2020-02-18 22:58:44.805  INFO 6624 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2020-02-18 22:58:44.831  INFO 6624 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-02-18 22:58:44.832  INFO 6624 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2020-02-18 22:58:44.837  INFO 6624 --- [           main] o.a.catalina.core.AprLifecycleListener   : Loaded APR based Apache Tomcat Native library [1.2.23] using APR version [1.7.0].
2020-02-18 22:58:44.837  INFO 6624 --- [           main]
```

```
        o.a.catalina.core.AprLifecycleListener   : APR capabilities: IPv6 [true], sendfile [true],
        accept filters [false], random [true].
156     2020-02-18 22:58:44.838  INFO 6624 --- [          main]
        o.a.catalina.core.AprLifecycleListener   : APR/OpenSSL configuration: useAprConnector
        [false], useOpenSSL [true]
157     2020-02-18 22:58:44.876  INFO 6624 --- [          main]
        o.a.catalina.core.AprLifecycleListener   : OpenSSL successfully initialized [OpenSSL 1.1.1c
         28 May 2019]
158     2020-02-18 22:58:45.105  INFO 6624 --- [          main]
        o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
        WebApplicationContext
159     2020-02-18 22:58:45.106  INFO 6624 --- [          main] o.s.web.context.ContextLoader
                : Root WebApplicationContext: initialization completed in 2557 ms
160     2020-02-18 22:58:45.589  INFO 6624 --- [          main]
        o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService
        'applicationTaskExecutor'
161     2020-02-18 22:58:46.146  INFO 6624 --- [          main]
        o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http)
        with context path ''
162     2020-02-18 22:58:46.157  INFO 6624 --- [          main]
        com.example.springbootdemo.App           : Started App in 4.718 seconds (JVM running
        for 7.46)
163
164   2)출력된 log 내용을 보면 8080 port로 tomcat이 시작된다는 것을 알 수 있다.
165   3)SpringApplication.run() method에서 내장 server를 시작했기 때문이다.
166   4)http://localhost:8080/로 접속해보자.
167   5)Web browser에 'Hello, World!'가 출력된다.
168   6)Console에는 아래와 같이 출력된다.
169     2020-02-18 23:00:42.429  INFO 6624 --- [nio-8080-exec-1]
        o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet
        'dispatcherServlet'
170     2020-02-18 23:00:42.429  INFO 6624 --- [nio-8080-exec-1]
        o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
171     2020-02-18 23:00:42.447  INFO 6624 --- [nio-8080-exec-1]
        o.s.web.servlet.DispatcherServlet        : Completed initialization in 18 ms
172
173   7)application을 끝내려면 Ctrl + C를 누르고, '[일괄 작업을 끝내시겠습니까 (Y/N)?'라는 질문에 'y'를 입력하고
      enter key를 누르면 된다.
174
175   8)여기서 알게 된 사실
176     -설정할 의존 관계의 갯수가 적다.
177     -Java Class 하나만 작성하면 된다.
178     -명령 prompt에서 application을 실행한다.
179
180
181 ----------------------------------------------------------------
182 Task2. STS로 Spring Boot Application 개발하기
183 1. Package Explorer > right-click > New > Spring Starter Project
184   1)Service URL :http://start.spring.io
185   2)Name : demo
186   3)Type : Maven
187   4)Packaging : jar
188   5)Java Version : 8
189   6)Language : Java
190   7)Group : com.example
191   8)Artifact : demo
192   9)Version : 0.0.1-SNAPSHOT
193   10)Description : Demo project for Spring Boot
```

```
   11)Package : com.example.demo
   12)Next


2. [New Spring Starter Project Dependencies] 창에서
   1)Spring Boot Version : 2.2.4
   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 ::        (v2.2.4.RELEASE)

   2020-02-18 23:07:20.979  INFO 5848 --- [           main]
   com.example.demo.DemoApplication         : Starting DemoApplication on
   DESKTOP-1BKHISM with PID 5848 (C:\SpringHome\demo\target\classes started by devex
   in C:\SpringHome\demo)
   2020-02-18 23:07:20.989  INFO 5848 --- [           main]
   com.example.demo.DemoApplication         : No active profile set, falling back to default
   profiles: default
   2020-02-18 23:07:23.442  INFO 5848 --- [           main]
   o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
   2020-02-18 23:07:23.467  INFO 5848 --- [           main]
   o.apache.catalina.core.StandardService   : Starting service [Tomcat]
   2020-02-18 23:07:23.468  INFO 5848 --- [           main]
   org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache
   Tomcat/9.0.30]
   2020-02-18 23:07:23.472  INFO 5848 --- [           main]
   o.a.catalina.core.AprLifecycleListener   : Loaded APR based Apache Tomcat Native library
   [1.2.23] using APR version [1.7.0].
   2020-02-18 23:07:23.473  INFO 5848 --- [           main]
   o.a.catalina.core.AprLifecycleListener   : APR capabilities: IPv6 [true], sendfile [true],
   accept filters [false], random [true].
   2020-02-18 23:07:23.473  INFO 5848 --- [           main]
   o.a.catalina.core.AprLifecycleListener   : APR/OpenSSL configuration: useAprConnector
   [false], useOpenSSL [true]
```

```
237   2020-02-18 23:07:23.483  INFO 5848 --- [           main] o.a.catalina.core.AprLifecycleListener   : OpenSSL successfully initialized [OpenSSL 1.1.1c 28 May 2019]
238   2020-02-18 23:07:23.710  INFO 5848 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
239   2020-02-18 23:07:23.711  INFO 5848 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 2584 ms
240   2020-02-18 23:07:24.174  INFO 5848 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
241   2020-02-18 23:07:24.703  INFO 5848 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
242   2020-02-18 23:07:24.713  INFO 5848 --- [           main] com.example.demo.DemoApplication         : Started DemoApplication in 4.801 seconds (JVM running for 7.357)
243
244
245 5. http://localhost:8080
246   Whitelabel Error Page
247   This application has no explicit mapping for /error, so you are seeing this as a fallback.
248   Wed Nov 06 20:02:04 KST 2019
249   There was an unexpected error (type=Not Found, status=404).
250   No message available
251
252
253 6. src/main/java/com.example.demo.DemoApplication.java 수정하기
254
255   package com.example.demo;
256
257   import org.springframework.boot.SpringApplication;
258   import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
259   import org.springframework.web.bind.annotation.RequestMapping;
260   import org.springframework.web.bind.annotation.RestController;
261
262   @RestController
263   @EnableAutoConfiguration
264   public class DemoApplication {
265
266     @RequestMapping("/")
267     String home() {
268       return "Hello, World!";
269     }
270
271     public static void main(String[] args) {
272       SpringApplication.run(DemoApplication.class, args);
273     }
274
275   }
276
277   1)DemoApplication.java > right-click > Run As > Spring Boot App
278   2)http://localhost:8080/
279     Hello, World!
280
281
282 ------------------------------------------------------------
283 Task3. Groovy로 Application 개발하기
284 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 저장한다.
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]
329     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]
330     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]
331     2020-02-18 23:22:22.800  INFO 12048 --- [      runner-0]
        o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
        WebApplicationContext
332     2020-02-18 23:22:22.801  INFO 12048 --- [      runner-0]
        o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization
        completed in 4158 ms
333     2020-02-18 23:22:23.454  INFO 12048 --- [      runner-0]
        o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService
        'applicationTaskExecutor'
334     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 ''
335     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)
336
337
338 3. Web browser로 http://localhost:8080에 접속한다.
339   Hello!!!
340
341
342 4. app.groovy code 수정하기
343   1)Command Prompt 에서 Ctrl + C를 눌러서 종료시킨다.
344   2)일괄 작업을 끝내시겠습니까 (Y/N)? Y
345   3)아래와 같이 code를 수정한다.
346
347     @RestController
348     class App {
349
350       @RequestMapping("/")
351       def home() {
352         def header = "<html><body>"
353         def footer = "</body></html>"
354         def content = "<h1>Hello! Spring Boot with Groovy</h1><p>This is html
          content.</p>"
355
356         header + content + footer
357       }
358     }
359
360
361 5. 다시 script를 실행한다.
362   $ spring run app.groovy
363
364
365 6. Browser를 refresh 한다.
366   Hello! Spring Boot with Groovy
367
```

368 This is html content.
369
370
371 7. Template 사용하기
372 1)template은 HTML을 기반으로 작성된 code를 읽어 rendering해서 web page에 출력하는 기능이다
373 2)이런 기능의 template이 몇 가지 종류가 있지만, spring boot에서는 thymeleaf(타임리프)라고 하는 library를 자주 사용한다.
374 3)http://www.thymeleaf.org
375 4)template file 작성
376 5)C:\temp\templates\home.html
377
378 <!doctype html>
379 <html lang="en">
380
381 <head>
382 <meta charset="UTF-8" />
383 <title>Index Page</title>
384 <style type="text/css">
385 h1 {
386 font-size: 18pt;
387 font-weight: bold;
388 color: gray;
389 }
390
391 body {
392 font-size: 13pt;
393 color: gray;
394 margin: 5px 25px;
395 }
396 </style>
397 </head>
398
399 <body>
400 <h1>Hello! Spring Boot with Thymeleaf</h1>
401 This is sample web page.
402 </body>
403
404 </html>
405
406 6)template file은 controller가 있는 곳의 templates folder 안에 두어야 한다.
407 7)controller 수정하기
408
409 -app.groovy
410 @Grab("thymeleaf-spring5")
411
412 @Controller
413 class App {
414
415 @RequestMapping("/")
416 @ResponseBody
417 def home(ModelAndView mav) {
418 mav.setViewName("home")
419 mav
420 }
421 }
422
423 8)다시 script 실행
424 $ 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)2.2.4
   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 : 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

      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"})
```

```
596     public class DemoApplication {
597       ...
598     }
599
600   2)com.example.vo package 생성
601   3)com.example.vo.UserVO Class 생성
602
603     package com.example.vo;
604
605     import lombok.Data;
606     import lombok.AllArgsConstructor;
607     import lombok.NoArgsConstructor;
608
609     @Data
610     @AllArgsConstructor
611     @NoArgsConstructor
612     public class UserVO {
613       private String userid;
614       private String name;
615       private String gender;
616       private String city;
617     }
618
619
620 10. UserController 생성하기
621   1)com.example.controller package 생성
622   2)com.example.controller.UserController Class 생성
623
624     package com.example.controller;
625
626     import org.springframework.web.bind.annotation.GetMapping;
627     import org.springframework.web.bind.annotation.RestController;
628
629     import com.example.vo.UserVO;
630
631     @RestController
632     public class UserController {
633       @GetMapping("/getUser")
634       public UserVO getUser() {
635         UserVO user = new UserVO();
636         user.setUserid("jimin");
637         user.setName("한지민");
638         user.setGender("여");
639         user.setCity("서울");
640         return user;
641       }
642     }
643
644   3)Relaunch demo
645     -http://localhost:8080/getUser
646     {"userid":"jimin","name":"한지민","gender":"여","city":"서울"}
647
648
649 11. Spring DevTools 사용하기
650   1)위처럼 Controller에 새로운 Method가 추가되면 반드시 실행 중인 Application을 중지하고 Application을 재
      실행해야 한다.
651   2)그렇게 해야만 수정된 Controller가 반영되기 때문이다.
652   3)그렇게 반복적인 작업을 하지 않고, 즉 Controller가 수정할 때마다 매번 Application을 재실행하는 것이 번거로
```

우면 Spring DevTools 기능을 이용하면 된다.
4)현재 사용중인 Project에 DevTools를 추가하려면 pom.xml에 추가 Dependency를 추가해야 한다.
5)pom.xml을 열어서 <dependency> 제일 마지막 Tag 밑에 Ctrl + Space 를 누른다.
6)Context Menu에서 [Edit Starters...]를 double-click한다.
7)[Pom file needs savings] 창에서 pom.xml 파일의 수정 내용을 저장할 것인지 물어보면 [Save Pom] button을 클릭한다.
8)[Edit Spring Boot Starters]창에서 Developer Tools > Spring Boot DevTools 체크한다.
9)그리고 [OK] 클릭한다.
10)그러면 pom.xml에 다음과 같은 Code가 추가된다

```
<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
public class UserVO {
  private String userid;
  private String name;
  private String gender;
  private String city;
}
```

23)수정된 UserVO.java 를 저장하고 왼쪽의 Package Explorer에서 UserVO.java의 하위를 클릭하면 Getter / Setter, toString() 이 자동으로 추가된 것을 확인할 수 있다.
24)다음은 Lombok에서 제공하는 Annotation이다.
-@Getter
-- Getter Method 생성
-@Setter
--Setter Method 생성
-@RequiredArgsConstructor
--모든 Member 변수를 초기화하는 생성자를 생성
-@ToString
--모든 Member 변수의 값을 문자열로 연결하여 리턴하는 toString() 메소드 생성
-@EqualsAndHashCode
--equals(), hashCode() Method 생성
-@Data
--@Getter, @Setter, @RequiredArgsConstructor, @ToString, @EquqlsAndHashCode 모두 생성.

Task5. SPRING INITIALIZR(Gradle)
1. Visit http://start.spring.io/
2. 설정
1)Gradle Project
2)Java
3)2.2.4
4)Group : com.example
5)Artifact : demoweb
6)Name : demoweb
7)Description : Demo project for Spring Boot
8)Package Name : com.example.demoweb
9)Packaging : Jar
10)Java Version : 8
11)dependencies : Developer Tools > SpringBoot DevTools, Web > Web

12)Click [Generate]
13)Downloads [demoweb.zip] : 57.8KB
14)Unpack to Spring workspace.

3. Project Import

1)In STS, Package Explorer > right-click > Import > Gradle > Existing Gradle Project > Next > Next
2)Click [Browse...] > demoweb Folder > Select Folder > Next > Finish
3)build.gradle

```
plugins {
  id 'org.springframework.boot' version '2.2.4.RELEASE'
  id 'io.spring.dependency-management' version '1.0.9.RELEASE'
  id 'java'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

configurations {
  developmentOnly
  runtimeClasspath {
    extendsFrom developmentOnly
  }
}

repositories {
  mavenCentral()
}

dependencies {
  implementation 'org.springframework.boot:spring-boot-starter-web'
  developmentOnly 'org.springframework.boot:spring-boot-devtools'
  testImplementation('org.springframework.boot:spring-boot-starter-test') {
    exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
  }
}

test {
  useJUnitPlatform()
}
```

4)src/test/java/com.example.demoweb.DemowebApplicationTests.java > right-click > Run As > JUnit Test > Green bar

5)demoweb Project > right-click > Run As > Spring Boot App
6)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 16:06:13 KST 2019
There was an unexpected error (type=Not Found, status=404).
No message available
```

4. Controller 생성
1)src/main/java/com.example.demoweb > right-click > New > Class
2)Name : HomeController

```
package com.example.demoweb;
```

```
816     import org.springframework.web.bind.annotation.GetMapping;
817     import org.springframework.web.bind.annotation.RestController;
818
819     @RestController
820     public class HomeController {
821
822       @GetMapping("/")
823       public String home() {
824         return "Hello, Spring Boot World";
825       }
826     }
827
828   3)Relaunch demo
829   4)http://localhost:8080/
830     Hello, Spring Boot World
831
832
833 ----------------------------------------------------------
834 Task6. 사용자 정의 Starter 만들기
835 1. Maven Project 생성
836   1)Project Explorer > right-click > New > Maven Project
837   2)Next
838   3)org.apache.maven.archetypes, maven-archetype-quickstart, 1.4 > Next
839   4)Group Id : com.example
840     -Artifact Id : mybootstarter
841     -Version : 0.0.1-SNAPSHOT
842     -Package : com.example.mybootstarter
843     -Finish
844
845
846 2. Project Facets 수정하기
847   1)mybootstarter Project > right-click > Properties > Project Facets
848   2)Java 1.8 > Runtimes Tab > Apache Tomcat v9.0, jdk1.8.0_241 check
849   3)Apply and Close click
850
851
852 3. Mvnrepository에서 'spring boot'로 검색
853   1)Spring Boot Autoconfigure > 2.2.4.RELEASE
854   2)아래 코드 복사 후 pom.xml 에 붙여넣기
855
856     <!--
      https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-autoconfigure
      -->
857     <dependency>
858       <groupId>org.springframework.boot</groupId>
859       <artifactId>spring-boot-autoconfigure</artifactId>
860       <version>2.2.4.RELEASE</version>
861     </dependency>
862
863   3)pom.xml > right-click > Run As > Maven install
864   [INFO] BUILD SUCCESS
865
866
867 4. dependencyManagement 추가
868   1)앞으로 추가되는 Library들의 Version을 일괄적으로 관리하기 위해 pom.xml에 Code 추가
869   2)Mvnrepository에서 'spring boot dependencies'로 검색
870   3)Spring Boot Dependencies에서 3.0.1.RELEASE
871   4)아래의 Code를 pom.xml의 제일 아래에 추가
```

```
872
873     <!--
        https://mvnrepository.com/artifact/org.apache.camel.springboot/camel-spring-boot-depe
        ndencies -->
874     <dependency>
875        <groupId>org.apache.camel.springboot</groupId>
876        <artifactId>camel-spring-boot-dependencies</artifactId>
877        <version>3.0.1</version>
878        <scope>provided</scope>
879     </dependency>
880
881   5)pom.xml
882
883     <dependencyManagement>
884       <dependencies>
885         <!--
            https://mvnrepository.com/artifact/org.apache.camel.springboot/camel-spring-boot-d
            ependencies -->
886         <dependency>
887           <groupId>org.apache.camel.springboot</groupId>
888           <artifactId>camel-spring-boot-dependencies</artifactId>
889           <version>3.0.1</version>
890           <type>pom</type>
891           <scope>provided</scope>
892         </dependency>
893       </dependencies>
894     </dependencyManagement>
895
896   6)project lombok library 추가
897     <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
898     <dependency>
899        <groupId>org.projectlombok</groupId>
900        <artifactId>lombok</artifactId>
901        <version>1.18.12</version>
902        <scope>provided</scope>
903     </dependency>
904
905   7)pom.xml > right-click > Run As > Maven install
906     [INFO] BUILD SUCCESS
907
908
909 5. 자동설정 구현하기
910   1)이번 실습은 JDBC로 직접 Database 연동을 처리하는 Application을 위한 자동 설정이 목표이다.
911   2)com.example.util package 생성
912   3)com.example.util.JdbcConnectionManager.java 구현
913
914     package com.example.util;
915
916     import java.sql.Connection;
917     import java.sql.DriverManager;
918
919     import lombok.Setter;
920     import lombok.ToString;
921
922     @Setter
923     @ToString
924     public class JdbcConnectionManager {
925       private String driverClass;
```

```
926         private String url;
927         private String username;
928         private String password;
929
930         public Connection getConnection() {
931           try {
932             Class.forName(this.driverClass);
933             return DriverManager.getConnection(this.url, this.username, this.password);
934           } catch (Exception e) {
935             e.printStackTrace();
936           }
937           return null;
938         }
939       }
940
941     4)JdbcConnectionManager를 bean으로 등록하는 환경 설정 Class를 작성한다.
942     5)com.example.config package 생성
943     6)com.example.config.UserAutoConfiguration.java 생성
944
945       package com.example.config;
946
947       import org.springframework.context.annotation.Bean;
948       import org.springframework.context.annotation.Configuration;
949
950       import com.example.util.JdbcConnectionManager;
951
952       @Configuration
953       public class UserAutoConfiguration {
954         @Bean
955         public JdbcConnectionManager getJdbcConnectionManager() {
956           JdbcConnectionManager manager = new JdbcConnectionManager();
957           manager.setDriverClass("oracle.jdbc.driver.OracleDriver");
958           manager.setUrl("jdbc:oracle:thin:@localhost:1521:XE");
959           manager.setUsername("hr");
960           manager.setPassword("hr");
961           return manager;
962         }
963       }
964
965
966   6. src/main/resources folder 추가
967     1)mybootstarter project > right-click > New > Source Folder
968     2)Folder name : src/main/resources
969     3)Finish
970
971
972   7. resources/META-INF folder 생성
973     1)src/main/resources > right-click > New > Folder
974     2)Folder name : META-INF
975     3)Finish
976
977
978   8. src/main/resources/META-INF/spring.factories file 생성
979     org.springframework.boot.autoconfigure.EnableAutoConfiguration=\com.example.config.Us
      erAutoConfiguration
980
981
982   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.

```
1140        Connection Manager : JdbcConnectionManager [driverClass=org.mariadb.jdbc.Driver,
1141        url=jdbc:mariadb://localhost:3306/test, username=root, password=javamariadb]
1142
1143    3)Property File 이용하기
1144      -Spring Container가 생성한 Bean의 Member Variable의 값이 자주 변경된다면 변경될 때마다 Java
          Source를 수정하기 보다 변경되는 정보만 Property로 등록하고 이 Property 정보를 이용해서 Bean을 생성하
          면 편리하다.
1145      -Lab을 위해서 demo Project의 com.example.config.UserConfiguration.java의 @Configuration
          과
1146
1147        @Bean을 주석처리한다.
1148        //@Configuration
1149        public class UserConfiguration {
1150
1151          //@Bean
1152          public JdbcConnectionManager getJdbcConnectionManager() {
1153            JdbcConnectionManager manager = new JdbcConnectionManager();
1154            manager.setDriverClass("org.mariadb.jdbc.Driver");
1155            manager.setUrl("jdbc:mariadb://localhost:3306/test");
1156            manager.setUsername("root");
1157            manager.setPassword("javamariadb");
1158            return manager;
1159          }
1160        }
1161
1162      -demo Project의 application.properites 파일에 다음 코드를 추가한다.
1163        ## Bean Overriding 설정
1164        spring.main.allow-bean-definition-overriding=true
1165        ## 데이터 소스 : Oracle
1166        user.jdbc.driverClass=oracle.jdbc.driver.OracleDriver
1167        user.jdbc.url=jdbc:oracle:thin:@localhost:1521:XE
1168        user.jdbc.username=hr
1169        user.jdbc.password=hr
1170
1171      -mybootstarter Project의 com.example.util.JdbcConnectionManagerProperties Class 추가로
          생성한다.
1172      -이미 만들어 놓은 com.example.util.JdbcConnectionManager.java를 복사, 붙여넣기, 이름변경을
          JdbcConnectionManagerProeprties로 한다.
1173
1174        package com.example.util;
1175
1176        import java.sql.Connection;
1177        import java.sql.DriverManager;
1178        import org.springframework.boot.context.properties.ConfigurationProperties;
1179
1180        @ConfigurationProperties(prefix="user.jdbc")
1181        public class JdbcConnectionManagerProperties {
1182          private String driverClass;
1183          private String url;
1184          private String username;
1185          private String password;
1186
1187          public String getDriverClass() {
1188            return driverClass;
1189          }
1190          public void setDriverClass(String driverClass) {
1191            this.driverClass = driverClass;
1192          }
```

```
1193          public String getUrl() {
1194            return url;
1195          }
1196          public void setUrl(String url) {
1197            this.url = url;
1198          }
1199          public String getUsername() {
1200            return username;
1201          }
1202          public void setUsername(String username) {
1203            this.username = username;
1204          }
1205          public String getPassword() {
1206            return password;
1207          }
1208          public void setPassword(String password) {
1209            this.password = password;
1210          }
1211        }
1212
1213      -위와 같이 작성하고 저장하면 노란색 경로라인이 발생한다.
1214      -노란색 경고 메시지에 마우스를 올려놓으면 Add spring-boot-configuration-processotr to pom.xml
          link를 click한다.
1215      -이렇게 하면 자동으로 pom.xml에 다음과 같은 dependency가 추가된다.
1216        <dependency>
1217          <groupId>org.springframework.boot</groupId>
1218          <artifactId>spring-boot-configuration-processor</artifactId>
1219          <optional>true</optional>
1220        </dependency>
1221
1222      -Error를 방지하기 위해 <version>2.2.0.RELEASE</version>을 추가한다.
1223      -mybootstarter Project > right-click > Maven > Update Project > OK
1224      -pom.xml > right-click > Run As > Maven install
1225        [INFO] BUILD SUCCESS
1226      -이제 mybootstarter Project의 com.example.config.UserAutoConfiguration Class를 수정한다.
1227
1228        package com.example.config;
1229
1230        import org.springframework.beans.factory.annotation.Autowired;
1231        import org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
1232        import org.springframework.boot.context.properties.EnableConfigurationProperties;
1233        import org.springframework.context.annotation.Bean;
1234        import org.springframework.context.annotation.Configuration;
1235        import com.example.util.JdbcConnectionManager;
1236        import com.example.util.JdbcConnectionManagerProperties;
1237
1238        @Configuration
1239        @EnableConfigurationProperties(JdbcConnectionManagerProperties.class)
1240        public class UserAutoConfiguration {
1241
1242          @Autowired
1243          private JdbcConnectionManagerProperties properties;
1244
1245          @Bean
1246          @ConditionalOnMissingBean
1247          public JdbcConnectionManager getJdbcConnectionManager() {
1248            JdbcConnectionManager manager = new JdbcConnectionManager();
1249            manager.setDriverClass(this.properties.getDriverClass());
```

```
            manager.setUrl(this.properties.getUrl());
            manager.setUsername(this.properties.getUsername());
            manager.setPassword(this.properties.getPassword());
            return manager;
          }
        }
```

-이제 mybootstarter Project를 다시 Install 한다.
  --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.

1305 --Driver Class : org.h2.Driver
1306 --JDBC URL : jdbc:h2:~/test
1307 --User Name : sa
1308 --Password : javah2
1309
1310 9)연결이 성공하면 Web Console이 열린다.
1311
1312
1313 2. JPA Project Installation
1314 1)In STS, Help > Install New Software
1315 2)Work with : https://download.eclipse.org/releases/2019-12
1316 3)Filter : jpa
1317 4)결과에서
1318 Web, XML, Java EE and OSGi Enterprise Development 하위의
1319 -Dali Java Persistence Tools - EclipseLink JPA Support
1320 -Dali Java Persistence Tools - JPA Diagram Editor
1321 -Dali Java Persistence Tools - JPA Support
1322 -m2e-wtp - JPA configurator for WTP (Optional)
1323
1324 5)설치 후 STS Restart
1325
1326 3. JPA Project 생성
1327 1)Package Explorer > right-click > Maven Project
1328 -Next
1329 -org.apache.maven.archetypes, maven-archetype-quickstart, 1.4
1330 -Next
1331
1332 2)각 항목 선택 후 Finish
1333 -Group Id : com.example
1334 -artifact Id : jpademo
1335 -Version : 0.0.1-SNAPSHOT
1336 -Package : com.example.jpademo
1337 -Finish
1338
1339 3)Project Facets 변환
1340 -jpademo Project > right-click > Properties > Project Facets
1341 -Java 1.8 > Runtimes > Apache Tomcat v9.0, jdk1.8.0_241 Check
1342 -Check JPA
1343 -만일 설정 화면 하단의 [Further configuration required...] Link에 Error message가 뜨는 경우
1344 --Link click
1345 --[JPA Facet] 창에서
1346 ---Platform : Generic 2.1
1347 ---JPA implementation
1348 Type : Disable Library Configuration
1349 --OK
1350 -Apply and Close
1351
1352 4)Maven Project를 JPA Project로 변경하면 src/main/java하위에 META-INF/persistence.xml JPA 환경설정 파일이 생긴다.
1353
1354 5)jpademo Project의 Perspective를 JPA Perspective로 변경하려면
1355 -Window > Perspective > Open Perspective > Other
1356 -JPA 선택 > Open
1357
1358
1359 4. 의존성 추가
1360 1)pom.xml을 수정
1361 -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;

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

/**
 * Entity implementation class for Entity: User
 *
 */
@Entity
@Table
@Getter
@Setter
@ToString
public class User implements Serializable {

  @Id
  private String userid;
  private String username;
  private String gender;
  private int age;
  private String city;

  private static final long serialVersionUID = 1L;

  public User() {
    super();
  }

}
```

11)다음은 JPA 사용하는 주요 Annotation을 설명한 것이다.

-@Entity
  --Entity Class 임을 설명
  --기본적으로 Class의 이름과 동일한 Table과 Mapping된다.
-@Table
  --Entity의 이름과 Table의 이름이 다를 경우, name 속성을 이용하여 Mapping 한다.
  --이름이 동일하면 생략 가능
-@Id
  --Table의 primary key와 Mapping한다.
  --Entiry의 필수 Annotation으로서 @Id가 없으면 Entity는 사용 불가
-@GeneratedValue
  --@Id가 선언된 Field에 기본 키 값을 자동으로 할당

6. JPA main 설정 파일 작성

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

```
<?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>
          <properties>
            <!-- 필수 속성 -->
            <property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>
            <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;

```

```
1760      public class JPAClient {
1761        public static void main(String[] args) {
1762          // EntityManager 생성
1763          EntityManagerFactory emf = Persistence.createEntityManagerFactory("jpademo");
1764          EntityManager em = emf.createEntityManager();
1765          // Transaction 생성
1766          EntityTransaction tx = em.getTransaction();
1767          try {
1768            // Transaction 시작
1769            tx.begin();
1770
1771            // 삭제할 User 조회
1772            User user = em.find(User.class, "younghee");
1773            em.remove(user);
1774
1775            // Transaction commit
1776            tx.commit();
1777          } catch (Exception e) {
1778            e.printStackTrace();
1779            // Transaction rollback
1780            tx.rollback();
1781          } finally {
1782            em.close();
1783            emf.close();
1784          }
1785        }
1786      }
1787
1788    2)실행
1789      Hibernate:
1790        select
1791          user0_.userid as userid1_0_0_,
1792          user0_.age as age2_0_0_,
1793          user0_.city as city3_0_0_,
1794          user0_.gender as gender4_0_0_,
1795          user0_.username as username5_0_0_
1796        from
1797          User user0_
1798        where
1799          user0_.userid=?
1800      Hibernate:
1801        delete
1802        from
1803          User
1804        where
1805
1806
1807  13. 여러 Record 조회와 JPQL
1808    1)한 건의 Record 조회는 find()를 사용한다.
1809    2)하지만, 여러 건의 Record를 조회하기 위해서는 JPQL(Java Persistence Query Language)라는 JPA에서
        제공하는 별도의 Query 명령어를 사용해야 한다.
1810    3)JPAClient.java 수정
1811
1812      import java.util.List;
1813
1814      import javax.persistence.EntityManager;
1815      import javax.persistence.EntityManagerFactory;
1816      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
        user0_.userid DESC
    User [userid=mija, username=이미자, gender=여, age=60, city=대구]
    User [userid=jimin, username=한지민, gender=여, age=24, city=서울]
    User [userid=hojune, username=이호준, gender=남, age=30, city=수원]
    User [userid=chulsu, username=김철수, gender=남, age=34, city=부산]


-------------------------------------------------------
Task8. 정적 Page 만들기
1. Spring Boot project 생성
  1)Package Explorer > right-click > New > Spring Starter Project
  2)다음 각 항목의 값을 입력한 후, Next 클릭한다.
    -Service URL :http://start.spring.io
    -Name : springweb
    -Type : Maven
    -Packaging : jar
    -Java Version : 8
    -Language : Java
    -Group : com.example
    -Artifact : springweb
    -Version : 0.0.1-SNAPSHOT
    -Description : Demo project for Spring Boot
    -Package : com.example.biz
    -Next

  3)다음의 각 항목을 선택한 후 Finish 클릭
    -Spring Boot Version : 2.2.4
    -Select Spring Web, Spring Boot DevTools


2. Controller 생성
  1)src/main/java/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;
    import org.springframework.web.servlet.ModelAndView;

    @Controller
    public class HomeController {
      @GetMapping("/")
      public ModelAndView home(ModelAndView mav) {
        mav.setViewName("index.html");
        return mav;
      }
    }


3. static file 생성
```

1)src/main/resources/static/images folder 생성
-spring-boot.png 추가할 것

2)src/main/resources/static/js folder 생성
-jquery-3.4.1.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-3.4.1.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 : 2.2.4
    -Select 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>9.0.27</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. com.example.biz.HomeController.java

```
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 02월 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 : 2.2.4
-Select Spring Web > Finish

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

```
package com.example.biz;

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

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

3. 실행
1)MyBootWebApplication.java > right-click > Run As > Spring Boot App
-http://localhost:8080/

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Fri Nov 08 23:25:37 KST 2019
There was an unexpected error (type=Not Found, status=404).
No message available

4. Controller 작성하기
1)com.example.biz > right-click > New > Class
2)Name : HelloController > Finish
3)HelloController.java

```
package com.example.biz;

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

@RestController
public class HelloController {

  @RequestMapping("/")
  public String index() {
    return "Hello Spring Boot World!";
  }
}
```

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

6. http://localhost:8080/

Hello Spring Boot World!

7. 매개변수 전달
  1)HelloController.java 수정

```
@RequestMapping("/{num}")
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) {
```

```
2220          return new Student(userid, names[userid], ages[userid], addresses[userid]);
2221        }
2222      }
2223
2224    3)http://localhost:8080/1
2225      {"userid":1,"name":"이미자","age":60,"address":"부산광역시"}
2226
2227
2228  11. thymeleaf 추가하기
2229    1)pom.xml 수정하기
2230      -Select Dependencies tab > Add
2231      -Group Id : org.springframework.boot
2232      -Artifact Id : spring-boot-starter-thymeleaf
2233      -Version :
2234      -Scope : compile
2235      -OK
2236
2237      <dependency>
2238        <groupId>org.springframework.boot</groupId>
2239        <artifactId>spring-boot-starter-thymeleaf</artifactId>
2240        <version>2.2.4.RELEASE</version>
2241      </dependency>
2242
2243    2)HelloController.java 수정
2244
2245      package com.example.biz;
2246
2247      import org.springframework.stereotype.Controller;
2248      import org.springframework.web.bind.annotation.RequestMapping;
2249
2250      @Controller
2251      public class HelloController {
2252
2253        @RequestMapping("/")
2254        public String index() {
2255          return "index";
2256        }
2257      }
2258
2259    3)template file 생성
2260      -src/main/resources/templates > right-click > New > Other...> Web > HTML File > Next
2261      -File name : index.html
2262
2263      <!doctype html>
2264      <html lang="en">
2265        <head>
2266          <meta charset="UTF-8" /> <!--반드시 종결 tag 필요 -->
2267          <title>Index Page</title>
2268          <style type="text/css">
2269            h1 { font-size:18pt; font-weight:bold; color:gray; }
2270            body { font-size:13pt; color:gray; margin:5px 25px; }
2271          </style>
2272        </head>
2273        <body>
2274          <h1>Hello! Spring Boot with Thymeleaf</h1>
2275          <p>This is sample web page.</p>
2276        </body>
2277      </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>
```

```
2336            <meta charset="UTF-8" />
2337            <title>Index Page</title>
2338            <style type="text/css">
2339              h1 { font-size:18pt; font-weight:bold; color:gray; }
2340              body { font-size:13pt; color:gray; margin:5px 25px; }
2341            </style>
2342          </head>
2343          <body>
2344            <h1>Hello!</h1>
2345            <p th:text="${msg}">${msg}</p>
2346            <form method="post" action="/send">
2347              <input type="text" name="txtName" th:value="${value}" />
2348              <input type="submit" value="Send" />
2349            </form>
2350          </body>
2351        </html>
2352
2353      -HelloController.java 수정
2354
2355        @Controller
2356        public class HelloController {
2357
2358          @RequestMapping(value = "/", method = RequestMethod.GET)
2359          public ModelAndView index(ModelAndView mav) {
2360            mav.setViewName("index");
2361            mav.addObject("msg", "Please write your name...");
2362            return mav;
2363          }
2364
2365          @RequestMapping(value = "/send", method = RequestMethod.POST)
2366          public ModelAndView send(@RequestParam("txtName") String name, ModelAndView
          mav) {
2367            mav.addObject("msg", "안녕하세요! " + name + "님!");
2368            mav.addObject("value", name);
2369            mav.setViewName("index");
2370            return mav;
2371          }
2372        }
2373
2374    10)http://localhost:8080
2375
2376    11)기타 form controller
2377      -index.html 수정
2378
2379      <!doctype html>
2380      <html lang="en">
2381        <head>
2382          <meta charset="UTF-8" />
2383          <title>Index Page</title>
2384          <style type="text/css">
2385            h1 {
2386              font-size: 18pt;
2387              font-weight: bold;
2388              color: gray;
2389            }
2390            body {
2391              font-size: 13pt;
2392              color: gray;
```

```
2393              margin: 5px 25px;
2394            }
2395          </style>
2396        </head>
2397        <body>
2398        <h1>Hello!</h1>
2399        <pre th:text="${msg}">Please wait...</pre>
2400        <form method="post" action="/">
2401        <div>
2402          <input type="checkbox" id="ckeck1" name="check1" /> <label for="ckeck1">체크
            </label>
2403        </div>
2404        <div>
2405          <input type="radio" id="male" name="gender" value="male" /> <label for="male">
            남성</label>
2406        </div>
2407        <div>
2408          <input type="radio" id="female" name="gender" value="female" /> <label
            for="female">여성</label>
2409        </div>
2410        <div>
2411          <select name="selOs" size="4">
2412            <option>--선택--</option>
2413            <option value="Windows">windows</option>
2414            <option value="MacOS">MacOS</option>
2415            <option value="Linux">Linux</option>
2416          </select>
2417          <select name="selEditors" size="4" multiple="multiple">
2418            <option>--선택--</option>
2419            <option value="Notepad">Notepad</option>
2420            <option value="Editplus">Editplus</option>
2421            <option value="Visual Studio Code">Visual Studio Code</option>
2422            <option value="Sublime Text">Sublime Text</option>
2423            <option value="Eclipse">Eclipse</option>
2424          </select>
2425        </div>
2426        <input type="submit" value="전송" />
2427        </form>
2428      </body>
2429    </html>
2430
2431    12)HelloController.java 수정
2432
2433      @Controller
2434      public class HelloController {
2435
2436        @RequestMapping(value = "/", method = RequestMethod.GET)
2437        public ModelAndView index(ModelAndView mav) {
2438
2439          mav.setViewName("index");
2440          mav.addObject("msg", "값을 입력후 전송버튼을 눌러주세요.");
2441          return mav;
2442        }
2443
2444        @RequestMapping(value = "/", method = RequestMethod.POST)
2445        public ModelAndView send(@RequestParam(value="check1", required=false) boolean
          check1,
2446              @RequestParam(value="gender", required=false) String gender,
```

```
            @RequestParam(value="selOs", required=false) String selOs,
            @RequestParam(value="selEditors", required=false) String [] selEditors,
            ModelAndView mav) {
        String result = "";
        try {
          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

```
2503 ------------------------------------------
2504 Task11. thymeleaf template 사용하기2
2505 1. Spring Boot project 생성
2506   1)Package Explorer > right-click > New > Spring Starter Project
2507
2508   2)다음 각 항목의 값을 입력 후 Next 클릭
2509     -Service URL :http://start.spring.io
2510     -Name : templatedemo
2511     -Type : Maven
2512     -Packaging : Jar
2513     -Java Version : 8
2514     -Language : Java
2515     -Group : com.example
2516     -Artifact : templatedemo
2517     -Version : 0.0.1-SNAPSHOT
2518     -Description : Demo project for Spring Boot
2519     -Package : com.example.biz
2520
2521   3)다음 각 항목 선택 후 Finish 클릭
2522     -Spring Boot Version : 2.2.1
2523     -Select
2524       --Developer Tools > Spring Boot DevTools, Lombok
2525       --Web > Spring Web
2526       --Template Engines > Thymeleaf
2527     -Finish
2528
2529
2530 2. HomeController.java 생성
2531   1)com.example.biz > right-click > New > Class
2532
2533   2)Name : HomeController
2534
2535     package com.example.biz;
2536
2537     import org.springframework.stereotype.Controller;
2538     import org.springframework.web.bind.annotation.GetMapping;
2539
2540     @Controller
2541     public class HomeController {
2542
2543       @GetMapping("/")
2544       public String home() {
2545         return "index";
2546       }
2547     }
2548
2549   3)index.html 생성
2550     -src/main/resources/templates > New > Other > Web > HTML File > Next
2551     -File name : index.html
2552
2553     <!DOCTYPE html>
2554     <html>
2555       <head>
2556         <meta charset="UTF-8" />
2557         <title>Insert title here</title>
2558       </head>
2559       <body>
2560         <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 때문이다.
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

```
  @RequestMapping("/")
  public ModelAndView index(ModelAndView mav) {
    mav.setViewName("index");
    mav.addObject("msg","Current data.");
    Member obj = new Member(123, "javaexpert",24);
    mav.addObject("object",obj);
    return mav;
  }
```

4)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; }
      </style>
    </head>
    <body>
      <h1 th:text="#{content.title}">Hello page</h1>
      <p th:text="${msg}">message.</p>
      <table th:object="${object}">
        <tr><th>ID</th><td th:text="*{id}"></td></tr>
        <tr><th>NAME</th><td th:text="*{username}"></td></tr>
        <tr><th>AGE</th><td th:text="*{age}"></td></tr>
      </table>
    </body>
  </html>
```

5)실행

6)controller에서 object를 저장해둔 Member 값이 표 형태로 출력된다

Task12. Spring Boot와 JDBC 연동하기

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 : BootJdbcDemo

-Type : Maven

-Packaging : Jar

-Java Version : 8

-Language : Java
-Group : com.example
-Artifact : BootJdbcDemo
-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 : 2.2.4
-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>9.0.27</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
```

```
2789
2790 4. Table 생성
2791   CREATE TABLE test.User
2792   (
2793     username VARCHAR(20) PRIMARY KEY,
2794     age TINYINT NOT NULL
2795   );
2796
2797
2798 5. VO object 생성
2799   1)src/main/java > right-click > New > Package
2800   2)Name : com.example.vo
2801   3)com.example.vo > right-click > New > Class
2802   4)Name : UserVO
2803
2804     package com.example.vo;
2805
2806     import lombok.Data;
2807     import lombok.AllArgsConstructor;
2808     import lombok.NoArgsConstructor;
2809
2810     @Data
2811     @AllArgsConstructor
2812     @NoArgsConstructor
2813     public class UserVO {
2814       private String username;
2815       private int age;
2816     }
2817
2818
2819 6. Dao object 생성
2820   1)src/main/java > right-click > New > Package
2821   2)Name : com.example.dao
2822   3)com.example.dao > right-click > New > Inteface
2823   4)Name : UserDao
2824
2825     package com.example.dao;
2826
2827     import java.util.List;
2828
2829     import com.example.vo.UserVO;
2830
2831     public interface UserDao {
2832       int create(UserVO userVO);
2833       List<UserVO> readAll();
2834       UserVO read(String username);
2835       int update(UserVO userVO);
2836       int delete(String username);
2837     }
2838
2839   5)com.example.dao > right-click > New > Class
2840   6)Name : UserDaoImpl
2841
2842     package com.example.dao;
2843
2844     import java.sql.ResultSet;
2845     import java.sql.SQLException;
2846     import java.util.List;
```

```
2847
2848    import org.springframework.beans.factory.annotation.Autowired;
2849    import org.springframework.jdbc.core.JdbcTemplate;
2850    import org.springframework.jdbc.core.RowMapper;
2851    import org.springframework.stereotype.Repository;
2852
2853    import com.example.vo.UserVO;
2854
2855    @Repository
2856    public class UserDaoImpl implements UserDao {
2857
2858      @Autowired
2859      private JdbcTemplate jdbcTemplate;
2860
2861      class UserMapper implements RowMapper<UserVO> {
2862        public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2863          UserVO user = new UserVO();
2864          user.setUsername(rs.getString("username"));
2865          user.setAge(rs.getInt("age"));
2866          return user;
2867        }
2868      }
2869
2870      @Override
2871      public int create(UserVO userVO) {
2872        String sql = "INSERT INTO User(username, age) VALUES(?, ?)";
2873        return this.jdbcTemplate.update(sql, userVO.getUsername(), userVO.getAge());
2874      }
2875
2876      @Override
2877      public List<UserVO> readAll() {
2878        String sql = "SELECT * FROM User";
2879        return this.jdbcTemplate.query(sql, new UserMapper());
2880      }
2881
2882      @Override
2883      public UserVO read(String username) {
2884        String sql = "SELECT * FROM User WHERE username = ?";
2885        return this.jdbcTemplate.queryForObject(sql, new Object[] {username}, new
          UserMapper());
2886      }
2887
2888      @Override
2889      public int update(UserVO userVO) {
2890        String sql = "UPDATE User SET age = ? WHERE username = ?";
2891        return this.jdbcTemplate.update(sql, userVO.getAge(), userVO.getUsername());
2892      }
2893
2894      @Override
2895      public int delete(String username) {
2896        String sql = "DELETE FROM User WHERE username = ?";
2897        return this.jdbcTemplate.update(sql, username);
2898      }
2899
2900    }
2901
2902
2903  7. Controller
```

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

    package com.example.biz;

    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.stereotype.Controller;
    import org.springframework.ui.Model;
    import org.springframework.web.bind.annotation.GetMapping;
    import org.springframework.web.bind.annotation.PostMapping;

    import com.example.dao.UserDao;
    import com.example.vo.UserVO;

    @Controller
    public class MainController {
      @Autowired
      private UserDao userDao;

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

      @PostMapping("/user")
      public String insert(UserVO userVO) {
        this.userDao.create(userVO);
        return "redirect:/user";
      }

      @GetMapping("/user")
      public String list(Model model) {
        model.addAttribute("users", this.userDao.readAll());
        return "list";
      }
    }


8. 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-3.4.1.min.js 추가할 것

  3)src/main/resources/static/css folder 생성
    -style.css

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


9. JSP를 위한 folder 준비
```

1)기본적으로 Spring Boot 에서는 jsp파일을 인식이 되지 않는다.
2)그래서 만일 jar로 packaging 한다면 embedded tomcat이 인식하는 web루트를 생성한다.
3)src > main 폴더 밑에 webapp와 jsp가 위치할 폴더를 만들어준다.
4)src/main folder 안에 webapp folder 생성
5)webapp folder 안에 WEB-INF folder 생성
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. 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() {}

  public MemberVO(String username, int age) {
    this.username = username;
    this.age = age;
  }

  public long getId() {
    return id;
  }

  public void setId(long id) {
    this.id = id;
```

```
3129        }
3130
3131        public String getUsername() {
3132          return username;
3133        }
3134
3135        public void setUsername(String username) {
3136          this.username = username;
3137        }
3138
3139        public int getAge() {
3140          return age;
3141        }
3142
3143        public void setAge(int age) {
3144          this.age = age;
3145        }
3146
3147        @Override
3148        public String toString() {
3149          return "MemberVO [id=" + id + ", username=" + username + ", age=" + age + "]";
3150        }
3151      }
3152
```

5)여기서 주의할 점은 기본 생성자는 반드시 넣어야 한다.

7. Repository

1)Entity class를 구성했다면 이번엔 Repository interface를 만들어야 한다.

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

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

4)Name : MemberRepository

```
3163      package com.example.biz;
3164
3165      import java.util.List;
3166
3167      import org.springframework.data.jpa.repository.Query;
3168      import org.springframework.data.repository.CrudRepository;
3169      import org.springframework.data.repository.query.Param;
3170
3171      public interface MemberRepository extends CrudRepository<MemberVO, Long> {
3172        List<MemberVO> findByUsernameAndAgeLessThan(String username, int age);
3173
3174        @Query("select t from Member t where username= :username and age < :age")
3175        List<MemberVO> findByUsernameAndAgeLessThanSQL(@Param("username") String
          username, @Param("age") int age);
3176
3177        List<MemberVO> findByUsernameAndAgeLessThanOrderByAgeDesc(String username,
          int age);
3178      }
3179
```

-위의 코드는 실제로 MemberVO Entity를 이용하기 위한 Repository class이다.

-기본적인 method 외에도 추가적인 method를 지정할 수 있다.

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

-findByUsernameAndAgeLessThan method와 findByUsernameAndAgeLessThanSQL method는 같은 결과를 출력하지만 전자의 method는 method 이름을 기반으로 한 것이고 후자의 method는 @Query annotation을 기반으로 해서 만든 것이다.
-method 이름 기반으로 해서 만들면 추후에 사용할 때 method 이름만으로도 어떤 query인지 알 수 있다는 장점이 있다.
-반대로 @Query annotation으로 만든 method는 기존의 source를 converting하는 경우 유용하게 사용할 수 있다.
-@Query
--다만 @Query annotation으로 query를 만들 때에 from 절에 들어가는 table은 Entity로 지정된 class 이름이다.
-method 이름 기반 작성법
-해당 부분은 Spring 문서 (https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.query-creation)를 통해 확인할 수 있다.

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

```
package com.example.biz;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class JpaDemoApplication implements CommandLineRunner {
  @Autowired
  MemberRepository memberRepository;

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

  @Override
  public void run(String... args) throws Exception {
    memberRepository.save(new MemberVO("a", 10));
    memberRepository.save(new MemberVO("b", 15));
    memberRepository.save(new MemberVO("c", 10));
    memberRepository.save(new MemberVO("a", 5));
    Iterable<MemberVO> list1 = memberRepository.findAll();
    System.out.println("findAll() Method.");
    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);
```

```
3231          for (MemberVO m : list3) {
3232            System.out.println(m.toString());
3233          }
3234
3235          System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
3236          List<MemberVO> list4 =
              memberRepository.findByUsernameAndAgeLessThanOrderByAgeDesc("a", 15);
3237          for (MemberVO m : list4) {
3238            System.out.println(m.toString());
3239          }
3240          memberRepository.deleteAll();
3241        }
3242
3243      }
3244
3245
3246  9. 실행
3247
3248    findAll() Method.
3249    MemberVO [id=1, username=a, age=10]
3250    MemberVO [id=2, username=b, age=15]
3251    MemberVO [id=3, username=c, age=10]
3252    MemberVO [id=4, username=a, age=5]
3253    findByUserNameAndAgeLessThan() Method.
3254    MemberVO [id=4, username=a, age=5]
3255    findByUserNameAndAgeLessThanSQL() Method.
3256    MemberVO [id=4, username=a, age=5]
3257    findByUserNameAndAgeLessThanSQL() Method.
3258    MemberVO [id=1, username=a, age=10]
3259    MemberVO [id=4, username=a, age=5]
3260
3261
3262  ----------------------------------------------------
3263  Task14. Spring Boot JPA
3264    1. Spring Boot project 생성
3265      1)Package Explorer > right-click > New > Spring Starter Project
3266
3267      2)다음의 각 항목의 값을 입력 후 Next 클릭
3268        -Service URL :http://start.spring.io
3269        -Name : BootJpaDemo
3270        -Type : Maven
3271        -Packaging : Jar
3272        -Java Version : 8
3273        -Language : Java
3274        -Group : com.example
3275        -Artifact : BootJpaDemo
3276        -Version : 0.0.1-SNAPSHOT
3277        -Description : Demo project for Spring Boot
3278        -Package : com.example.biz
3279
3280      3)각 항목을 선택 후 Finish 클릭
3281        -Spring Boot Version : 2.2.4
3282        -Select
3283          --SQL > Spring Data JPA, H2 Database
3284          --Developer Tools > Spring Boot DevTools
3285          --Web > Spring Web
3286          --Template Engines > Thymeleaf
3287        -Finish
```

4)DevTools
-It provides developer tools.
-These tools are helpful in application development mode.
-One of the features of developer tool is automatic restart of the server for any change in code

2. Entity 작성하기
1)com.example.biz > right-click > New > Package
2)Name : com.example.biz.vo
3)com.example.biz.vo > right-click > New > Class
4)Name : User

```
package com.example.biz.vo;

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

import lombok.Data;

@Entity
@Table
public class User {

  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  @Column
  private long id;

  @Column(length = 20, nullable = false)
  private String username;

  @Column(length = 100, nullable = true)
  private String email;

  @Column(nullable = true)
  private Integer age;

  @Column(nullable = true)
  private String memo;

  public long getId() {
    return id;
  }

  public void setId(long id) {
    this.id = id;
  }

  public String getUsername() {
    return username;
  }

```

```
3345        public void setUsername(String username) {
3346          this.username = username;
3347        }
3348
3349        public String getEmail() {
3350          return email;
3351        }
3352
3353        public void setEmail(String email) {
3354          this.email = email;
3355        }
3356
3357        public Integer getAge() {
3358          return age;
3359        }
3360
3361        public void setAge(Integer age) {
3362          this.age = age;
3363        }
3364
3365        public String getMemo() {
3366          return memo;
3367        }
3368
3369        public void setMemo(String memo) {
3370          this.memo = memo;
3371        }
3372      }
3373
3374
3375  3. Repository 생성하기
3376    1)com.example.biz > right-click > New > Package
3377    2)Name : com.example.biz.dao
3378    3)com.example.biz.dao > right-click > New > Interface
3379    4)Name : UserRepository > Finish
3380
3381      package com.example.biz.dao;
3382
3383      import org.springframework.data.jpa.repository.JpaRepository;
3384      import org.springframework.stereotype.Repository;
3385
3386      import com.example.biz.vo.User;
3387
3388      @Repository("repository")
3389      public interface UserRepository extends JpaRepository<User, Long>{
3390
3391      }
3392
3393      -JpaRepository라는 interface는 새로운 repository를 생성하기 위한 토대가 된다.
3394      -모든 Repository는 이 JpaRepository를 상속해서 작성한다
3395
3396
3397  4. HelloController 작성
3398    1)com.example.biz > right-click > New > Class
3399    2)Name : HelloController
3400
3401      package com.example.biz;
3402
```

```
3403      import org.springframework.beans.factory.annotation.Autowired;
3404      import org.springframework.stereotype.Controller;
3405      import org.springframework.web.bind.annotation.RequestMapping;
3406      import org.springframework.web.servlet.ModelAndView;
3407
3408      import com.example.biz.dao.UserRepository;
3409      import com.example.biz.vo.User;
3410
3411      @Controller
3412      public class HelloController {
3413
3414        @Autowired
3415        UserRepository repository;
3416
3417        @RequestMapping("/")
3418        public ModelAndView index(ModelAndView mav) {
3419          mav.setViewName("index");
3420          mav.addObject("msg", "this is sample content.");
3421          Iterable<User> list = repository.findAll();
3422          mav.addObject("data", list);
3423          return mav;
3424        }
3425      }
3426
3427      -UserRepository에는 findAll 같은 method가 정의되어 있지 않다.
3428      -이것은 부모 interface인 JpaRepository가 가지고 있는 method이다.
3429      -이를 통해 모든 entity가 자동으로 추출되는 것이다.
3430
3431
3432  5. JUnit Test
3433    1)src/test/java/com.example.biz.BootJpaDemoApplicationTests.java > right-click > Run As
       > JUnit Test
3434    2)Green bar
3435
3436
3437  6. template 준비하기
3438    1)src/main/resources > right-click > New > File
3439
3440    2)File name : messages.properties
3441      content.title=Message sample page.
3442      content.message=This is sample message from properties.
3443
3444    3)src/main/resources/templates > right-click > New > Web > HTML Files
3445    4)Name : index.html
3446
3447      <!DOCTYPE HTML>
3448      <html xmlns:th="http://www.thymeleaf.org">
3449        <head>
3450          <title>top page</title>
3451          <meta charset="UTF-8" />
3452          <style>
3453            h1 { font-size:18pt; font-weight:bold; color:gray; }
3454            body { font-size:13pt; color:gray; margin:5px 25px; }
3455            pre { border: solid 3px #ddd; padding: 10px; }
3456          </style>
3457        </head>
3458        <body>
3459          <h1 th:text="#{content.title}">Hello page</h1>
```

```
3460          <pre th:text="${data}"></pre>
3461        </body>
3462      </html>
3463
3464
3465  7. 실행해서 접속
3466    1)저장돼있는 data가 회색 사각 틀 안에 표시된다.
3467    2)아직 아무 data가 없기 때문에 빈 배열 []라고 표시된다.
3468
3469
3470  8. Entity의 CRUD 처리하기 : form으로 data 저장하기
3471    1)index.html 수정
3472
3473      <!DOCTYPE HTML>
3474      <html xmlns:th="http://www.thymeleaf.org">
3475      <head>
3476      <title>top page</title>
3477      <meta charset="UTF-8" />
3478      <style>
3479      h1 {
3480        font-size: 18pt;
3481        font-weight: bold;
3482        color: gray;
3483      }
3484
3485      body {
3486        font-size: 13pt;
3487        color: gray;
3488        margin: 5px 25px;
3489      }
3490
3491      tr {
3492        margin: 5px;
3493      }
3494
3495      th {
3496        padding: 5px;
3497        color: white;
3498        background: darkgray;
3499      }
3500
3501      td {
3502        padding: 5px;
3503        color: black;
3504        background: #e0e0ff;
3505      }
3506      </style>
3507      </head>
3508      <body>
3509        <h1 th:text="#{content.title}">Hello page</h1>
3510        <table>
3511          <form method="post" action="/" th:object="${formModel}">
3512            <tr>
3513              <td><label for="username">이름</label></td>
3514              <td><input type="text" name="username" th:value="*{username}" /></td>
3515            </tr>
3516            <tr>
3517              <td><label for="age">연령</label></td>
```

```
3518              <td><input type="text" name="age" th:value="*{age}" /></td>
3519            </tr>
3520            <tr>
3521              <td><label for="email">메일</label></td>
3522              <td><input type="text" name="email" th:value="*{email}" /></td>
3523            </tr>
3524            <tr>
3525              <td><label for="memo">메모</label></td>
3526              <td><textarea name="memo" th:text="*{memo}" cols="20"
                  rows="5"></textarea></td>
3527            </tr>
3528            <tr>
3529              <td></td>
3530              <td><input type="submit" /></td>
3531            </tr>
3532          </form>
3533        </table>
3534        <hr />
3535        <table>
3536          <tr>
3537            <th>ID</th>
3538            <th>이름</th>
3539          </tr>
3540          <tr th:each="obj : ${datalist}">
3541            <td th:text="${obj.id}"></td>
3542            <td th:text="${obj.username}"></td>
3543          </tr>
3544        </table>
3545      </body>
3546      </html
3547
3548    2)HelloController.java 수정
3549
3550      package com.example.biz;
3551
3552      import org.springframework.beans.factory.annotation.Autowired;
3553      import org.springframework.stereotype.Controller;
3554      import org.springframework.transaction.annotation.Transactional;
3555      import org.springframework.web.bind.annotation.ModelAttribute;
3556      import org.springframework.web.bind.annotation.RequestMapping;
3557      import org.springframework.web.bind.annotation.RequestMethod;
3558      import org.springframework.web.servlet.ModelAndView;
3559
3560      import com.example.biz.dao.UserRepository;
3561      import com.example.biz.vo.User;
3562
3563      @Controller
3564      public class HelloController {
3565
3566        @Autowired
3567        UserRepository repository;
3568
3569        @RequestMapping(value = "/", method = RequestMethod.GET)
3570        public ModelAndView index(@ModelAttribute("formModel") User mydata, ModelAndView
          mav) {
3571          mav.setViewName("index");
3572          mav.addObject("msg", "this is sample content.");
3573          Iterable<User> list = repository.findAll();
```

```
3574          mav.addObject("datalist", list);
3575          return mav;
3576        }
3577
3578        @RequestMapping(value = "/", method = RequestMethod.POST)
3579        @Transactional(readOnly = false)
3580        public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
            mav) {
3581          repository.saveAndFlush(mydata);
3582          return new ModelAndView("redirect:/");
3583        }
3584      }
3585
3586
3587  9. @ModelAttribute와 data 저장
3588    1)@ModelAttribute
3589      -이것은 entity class의 instance를 자동으로 적용할 때 사용
3590      -인수에는 instance 이름을 지정한다.
3591      -이것은 전송 form에서 th:object로 지정하는 값이 된다.
3592      -전송된 form의 값이 자동으로 User instance로 저장된다.
3593      -따라서 이 annotation을 이용하면 이렇게 쉽게 전송한 data를 저장할 수 있다.
3594
3595    2)saveAndFlush() method
3596      -HomeController.java의 아래 code를 보자.
3597        @RequestMapping(value = "/", method = RequestMethod.POST)
3598        @Transactional(readOnly=false)
3599        public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
            mav) {
3600          repository.saveAndFlush(mydata);
3601          return new ModelAndView("redirect:/");
3602        }
3603
3604    3)미리 설정한 entity는 JpaRepository의 saveAndFlush라는 method를 통해 entity를 영구화한다.
3605    4)Database를 사용하고 있다면 Database에 그대로 저장된다
3606
3607
3608  10. @Transactional과 transaction
3609    1)바로 위의 code에서 @Transactional(readOnly=false)가 있다.
3610    2)이 annotation은 transaction을 위한 것이다.
3611    3)이 annotation때문에 method내에서 실행되는 database 처리가 일괄적으로 실행되게 된다.
3612    4)data 변경 처리는 도중에 외부 접속에 의해 data 구조나 내용이 바뀌면 data 일관성에 문제가 발생하게 된다.
3613    5)이런 문제를 방지하기 위해 transaction이 사용되는 것이다
3614    6)code를 보면 readOnly=false라고 설정하고 있다.
3615    7)이 readOnly는 문자 그대로 '읽기 전용(변경 불가)임을 의미한다.
3616    8)readOnly=false라고 설정하면 변경을 허가하는 transaction이다.
3617
3618
3619  11. Data 초기화 처리
3620    1)저장한 data는 application을 종료하고 다시 실행하면 지워진다.
3621    2)HSQLDB는 기본적으로 memory내에 data를 cache하고 있으므로 종료와 함께 지워지는 것이다.
3622    3)controller에 data를 작성하는 초기화 처리를 넣기로 한다.
3623    4)HelloController.java code 추가
3624
3625      @PostConstruct
3626      public void init(){
3627        User user1 = new User();
3628        user1.setUsername("한지민");
3629        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;
    }

    tr {
        margin: 5px;
    }

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

    td {
```

```
3687        padding: 5px;
3688        color: black;
3689        background: #e0e0ff;
3690      }
3691      </style>
3692      </head>
3693      <body>
3694        <h1 th:text="${title}">Edit page</h1>
3695        <table>
3696          <form method="post" action="/edit" th:object="${formModel}">
3697            <input type="hidden" name="id" th:value="*{id}" />
3698            <tr>
3699              <td><label for="username">이름</label></td>
3700              <td><input type="text" name="username" th:value="*{username}" /></td>
3701            </tr>
3702            <tr>
3703              <td><label for="age">연령</label></td>
3704              <td><input type="text" name="age" th:value="*{age}" /></td>
3705            </tr>
3706            <tr>
3707              <td><label for="email">메일</label></td>
3708              <td><input type="text" name="email" th:value="*{email}" /></td>
3709            </tr>
3710            <tr>
3711              <td><label for="memo">메모</label></td>
3712              <td><textarea name="memo" th:text="*{memo}" cols="20"
                  rows="5"></textarea></td>
3713            </tr>
3714            <tr>
3715              <td></td>
3716              <td><input type="submit" /></td>
3717            </tr>
3718          </form>
3719        </table>
3720      </body>
3721      </html>
3722
3723    3)UserRepository code 추가
3724
3725      package com.example.biz.dao;
3726
3727      import java.util.Optional;
3728
3729      import org.springframework.data.jpa.repository.JpaRepository;
3730      import org.springframework.stereotype.Repository;
3731
3732      import com.example.biz.vo.User;
3733
3734      @Repository("repository")
3735      public interface UserRepository extends JpaRepository<User, Long>{
3736        public Optional<User> findById(Long id);
3737      }
3738
3739    4)RequestHandler 작성하기
3740      -HelloController.java code 추가
3741
3742        @RequestMapping(value = "/edit/{id}", method = RequestMethod.GET)
3743        public ModelAndView edit(@ModelAttribute User user, @PathVariable int id,
```

```
        ModelAndView mav) {
          mav.setViewName("edit");
          mav.addObject("title", "edit mydata.");
          Optional<User> findUser = repository.findById((long) id);
          mav.addObject("formModel", findUser.get());
          return mav;
        }

        @RequestMapping(value = "/edit", method = RequestMethod.POST)
        @Transactional(readOnly = false)
        public ModelAndView update(@ModelAttribute User user, ModelAndView mav) {
          repository.saveAndFlush(user);
          return new ModelAndView("redirect:/");
        }
```

5)접속해서 아래와 같이 URL을 입력하면

http://localhost:8080/edit/1

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

13. Entity delete 구현하기
1)update를 하고 select를 했으니 이번에는 delete를 해 보자.
2)delete.html template를 작성한다.
3)src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
4)File name : delete.html > Finish

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

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

td {
  padding: 0px 20px;
  background: #eee;
}
</style>
</head>
<body>
  <h1 th:text="${title}">Delete page</h1>
```

```
      <table>
        <form method="post" action="/delete" th:object="${formModel}">
          <input type="hidden" name="id" th:value="*{id}" />
          <tr>
            <td><p th:text="|이름 :  *{username}|"></p></td>
          </tr>
          <tr>
            <td><p th:text="|연령 :  *{age}|"></p></td>
          </tr>
          <tr>
            <td><p th:text="*{email}"></p></td>
          </tr>
          <tr>
            <td><p th:text="*{memo}"></p></td>
          </tr>
          <tr>
            <td><input type="submit" value="delete" /></td>
          </tr>
        </form>
      </table>
    </body>
    </html>
```

5)RequestHandler 작성

```
  @RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
    public ModelAndView delete(@PathVariable int id, ModelAndView mav) {
      mav.setViewName("delete");
      mav.addObject("title", "delete mydata.");
      Optional<User> user = repository.findById((long) id);
      mav.addObject("formModel", user.get());
      return mav;
    }

    @RequestMapping(value = "/delete", method = RequestMethod.POST)
    @Transactional(readOnly = false)
    public ModelAndView remove(@RequestParam long id, ModelAndView mav) {
      repository.deleteById(id);
      return new ModelAndView("redirect:/");
    }
```

6)접속해서 실행

http://localhost:8080/delete/2라고 하면 id가 2번이 출력되고 delete button을 누르면 삭제된다.